An Efficient SAT Encoding of Circuit Codes

Yury CHEBIRYAK† and Daniel KROENING‡

† Computer Systems Institute
ETH Zurich
8092 Zurich, Switzerland
E-mail: yury.chebiryak@inf.ethz.ch

‡ Computing Laboratory
Oxford University
Oxford, OX1 3QD, UK
E-mail: kroening@comlab.ox.ac.uk

Abstract

Circuit codes in hypercubes are generalized snake-in-the-box codes and are used in analog-to-digital conversion devices. 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. In this paper, we describe an efficient SAT encoding of circuit codes, which enabled us to obtain new circuit codes.

1. INTRODUCTION

In 1958, W. H. Kautz brought attention to the snake-in-the-box problem—finding a binary code that has unit distance between adjacent code words and minimum distance two between all other code words [14]. The search for snakes is motivated by the theory of error-correcting codes (as the vertices of a solution to the snake or coil in the box problems can be used as a Gray code that can detect single-bit errors), electrical engineering, computer network topologies [1], Systems Biology [9], etc. Approaches to find long snakes range from studies of mathematical constructions (e.g. binary necklaces [17]) and certain patterns in lower dimensions [20, 19] to genetic algorithms [1, 2, 22].

R. C. Singleton generalized the concept of snake-in-the-box codes to circuit codes with a parameter spread [21]. A circuit code of spread δ has unit distance between adjacent code words, and minimum distance δ between code words δ apart in the ordered sequence. For example, the circuit codes with the spread δ = 2 are the coil-in-the-box, and the codes with δ = 1 and 2^n distinct code words are the Hamiltonian cycles of the n-cube. Singleton then presented constructions for circuit codes for spreads up to 7.

Circuit codes are useful in correcting and limiting errors in analog-to-digital conversion (see [16]). The longer the code, the greater the accuracy of the system (while the greater the spread, the greater the error-detection capability). Therefore, determining the length of the longest n-dimensional circuit code of spread δ is of interest [8, 23].

V. Klee showed the construction of a code with even spread δ by extending a code of spread δ using a code of spread δ − 1 [15]. K. Deimer described a method for finding a circuit code of spread δ and length L − k in the n-cube from a circuit code in dimension n + 1 of spread δ + 1 and length L [4]. Here k is the number of times a certain transition is taken (this transition number is then removed from the transition sequence). It is not evident that finding such a code (of higher spread and length, in higher dimension) is easier than the target circuit code. Paterson and Tuliani presented a construction method based on binary necklaces [17], generalizing ideas for obtaining single-track circuit codes of [7]. In earlier work, we improved lower bounds and proved optimality of circuit codes for 14 different sets of parameters (n, δ) [24]. The approach uses a SAT-solver and is not limited to specific values of a spread.

In this paper, we present new (longer than previously known) circuit codes that we have obtained using a novel efficient propositional satisfiability encoding.

2. CIRCUIT CODES

Consider an ordered sequence C of L binary code words \(W_0, W_1, \ldots, W_{L-1}\). Let \(d^k_m\) denote the Hamming distance between code words \(W_k\) and \(W_m\):

\[ d^k_m = |\{ i \mid W_k[i] \neq W_m[i] \}| , \]

where \(W_k[i]\) denotes the i-th bit of k-th code word, and let \(d_C\) be the cyclic distance between code words in the sequence [12]:

\[ d_C^k_m = \min(|k - m|, L - |k - m|) . \]

A path in a hypercube is a sequence of code words, in which consecutive elements have unit Hamming dis-
In a cycle, the first and the last code words also have unit Hamming distance:

\[ \forall k \in \{0, 1, \ldots, L\} : \quad d_{C}^{k,k+1} \mod L = 1 \implies d_{H}^{k,k+1} \mod L = 1. \]

The definitions of circuit codes by Paterson and Tuliani [17] and by Preparata and Nievergelt [18] differ slightly, but were proven equivalent by L. Haryanto [12].

Definition 1 (Circuit code [13]). A length \( L \), spread \( \delta \) circuit code in the \( n \)-cube (or \((n, L, \delta)\)-CC) is a cyclic path \( C \) of \( L \) binary \( n \)-tuples \( W_{0}, W_{1}, \ldots, W_{L-1} \) with the property that for all \( k, m \in \{0, 1, \ldots, L-1\}, \)

\[ d_{H}^{k,m} < \delta \implies d_{C}^{k,m} < \delta. \]  

(1)

3. PROPOSITIONAL SAT ENCODING

A SAT solver determines whether a propositional formula, given in conjunctive normal form (CNF), is satisfiable. If it is, the solver provides a satisfying assignment to the variables in the formula.

In this section, we describe our encoding of a search for circuit codes into a propositional SAT formula in detail. Then, we improve it using an observation about the circuit codes’ structure and obtain new codes.

3.1. The Satisfiability Problem

Let \( V = \{x_{0}, x_{1}, \ldots, x_{L-1}\} \) be a set of Boolean variables. A literal is a variable \( x_{i} \) or its negation \( \neg x_{i} \). Let \( \phi \) be a propositional formula over the variables in \( V \). The propositional satisfiability (SAT) problem [10] is to determine whether there exists an assignment of truth values to the variables in \( V \) such that the formula \( \phi \) evaluates to true.

3.2. Encoding of Circuit Codes

Our goal is to construct a formula with satisfying assignments corresponding to the coordinates of the nodes forming a spread-\( \delta \) circuit code of \( L \) code words in the \( n \)-cube. For this purpose, we define \( n \)-\( L \) Boolean variables denoted by \( W_{i}[j] \), where \( i \in \{0, \ldots, L-1\} \) and \( j \in \{0, \ldots, n-1\} \). The Boolean vector \( W_{i} \) denotes the coordinates of node number \( i \) of the code, where \( W_{i}[0] \) corresponds to the right-most bit of the coordinates of node \( W_{i} \).

For a sequence of nodes \( W_{0}, W_{1}, \ldots, W_{L-1} \), we encode the following constraints:

1. To form a cycle in an \( n \)-cube, the neighbouring nodes of a sequence must be adjacent. The adjacency is expressed using the Hamming distance:

\[ \phi_{\text{cycle}} := \bigwedge_{k=0}^{L-1} (d_{H}^{k,k+1} \mod L = 1). \]  

(2)

2. The formula (1) suggests to pick pairs of code-words \( W_{k} \) and \( W_{m} \), that are at least \( \delta \) apart in the sequence\(^1\) and require their Hamming distances to be at least \( \delta \):

\[ \phi_{\delta} := \bigwedge_{0 \leq k < m < L} (d_{C}^{k,m} \geq \delta \implies d_{H}^{k,m} \geq \delta). \]  

(3)

The propositional formula

\[ \phi_{\text{CC}} := \phi_{\text{cycle}} \land \phi_{\delta} \]  

(4)

encodes an \((n, L, \delta)\)-CC. A satisfying assignment of \( \phi_{\text{CC}} \) contains the coordinates of some circuit code with these parameters.

We encoded formulae (2) and (3) using once-twice chains and bitonic sorting networks respectively (for details, see [3]) and obtained new circuit codes in [24].

3.3. A More Efficient Encoding

One can reduce the number of variables and clauses by a factor of two.

Consider nodes \( W_{k} \) and \( W_{m} \) together with neighbours of \( W_{m-1} \) and \( W_{m+1} \). If \( W_{k} \) is at least \( \delta \) apart in the sequence from each of these three nodes, we have to require that their Hamming distances are at least \( \delta \) each (see Figure 1).

Suppose that \( k \) and \( m \) are of the same parity, i.e. the distance \( d_{C}^{k,m} \) is even. Then, by 2-colourability of a hypercube [11], the Hamming distance is also even. If the value of the spread is odd, we can strengthen the property (3) to \( d_{H}^{k,m} \geq \delta + 1 \), which implies the separability condition for \( W_{k} \) and neighbours of \( W_{m} \) (because Hamming distances for them may decrease only by one). In Eq. (3) we can therefore reduce the number of constraints by about one half due to redundancy.

3.4. Evaluation

\(^1\)With such a formulation, for codes with higher spreads there are fewer of pairs to consider (given that dimension and length are fixed), hence an encoding of these codes requires fewer variables and clauses. This is advantageous as performance of existing approaches decays with increasing spread (e.g., the construction in [17] uses special kinds of binary necklaces and finding them for higher spreads is hard).
We generalize this observation, by modifying the Eq. (3) as follows:

$$\phi' := \bigwedge_{0 \leq k < m < L} \left( \left( d_{C}^{k,m} \mod 2 \neq \delta \mod 2 \right) \land d_{C}^{k,m} \geq \delta \Rightarrow d_{H}^{k,m} \geq \delta + 1 \right). \quad (5)$$

The modified encoding of a (10,84,3)-CC takes 49.9% less variables and clauses. The runtime² for this instance is decreased by 51.5%.

Using the efficient encoding we obtained 11 new circuit codes (see Table 1).

References


²For our experiments, we use the MiniSat, written by Eén and Sörensson [6]. MiniSat provides interfaces for incremental solving and All-SAT. The current version uses preprocessing techniques from QBF-Solvers [5]. All experiments were carried out on a PC with an Intel Xeon (3.0-GHz, 4-GB RAM, running Linux) with a timeout of 24 h.
Table 1: New Circuit codes

<table>
<thead>
<tr>
<th>n</th>
<th>L</th>
<th>δ</th>
<th>Transition sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>86</td>
<td>3</td>
<td>461328167a3869a3495769a4629832173497531a5761547852793487591a754a615a8624973526758a1687</td>
</tr>
<tr>
<td>11</td>
<td>68</td>
<td>4</td>
<td>546138b5497b6521b79a5b14985b279543268597b213879245a836b29a6872a4b629</td>
</tr>
<tr>
<td>12</td>
<td>98</td>
<td>4</td>
<td>18523b95a461b89a175321bc42385928cba91c83b59c16b748391ca596342b53a6174a381ba58eb23891a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b237e4a82bca</td>
</tr>
<tr>
<td>12</td>
<td>56</td>
<td>5</td>
<td>8ca5798a473b91a547218ca5b143a79b463ca54732ce8b31279a342</td>
</tr>
<tr>
<td>13</td>
<td>50</td>
<td>6</td>
<td>1b264a98b65da39715be49a18b246aad5b1c4a89d73cba86d5c</td>
</tr>
<tr>
<td>14</td>
<td>66</td>
<td>6</td>
<td>85d473c2b517e94362d795384c719da64731b5429618349e157e843b615294bac</td>
</tr>
<tr>
<td>15</td>
<td>82</td>
<td>6</td>
<td>d4c61b9dc3f7bad436785ab23e18579a4df8295e48c19da68eb94236dfe9267ba5d293f4bde8369a2</td>
</tr>
<tr>
<td>16</td>
<td>106</td>
<td>6</td>
<td>gdbc146g7dec234gdefbc157af48637ce519b7dcfg94a52b713f92db63af827b1ef8379b6f48dgbef6c9854be1c7ag452b368f5492</td>
</tr>
<tr>
<td>16</td>
<td>72</td>
<td>7</td>
<td>b951g7e2b81d5c4ef238a147d92561ebf48276gf31d47a26985d7efac4b9de8a2713d6e</td>
</tr>
<tr>
<td>17</td>
<td>90</td>
<td>7</td>
<td>472h8g13f7e2d5gh9c762efa8cg697bh3fgdc4723659e1ffda259b73fha5e968b274ce9d561fc7ae981ge5b69</td>
</tr>
<tr>
<td>17</td>
<td>64</td>
<td>8</td>
<td>fc47e8139b5gca7bfehd456738eh9fcg4213e5fb94dg7ehf38bac9ge5682b</td>
</tr>
</tbody>
</table>


