

Towards a Classification of Hamiltonian Cycles in the 6-Cube

Yury Chebiryak*

*Computer Systems Institute,
ETH Zurich,
8092 Zurich, Switzerland.*

yury.chebiryak@inf.ethz.ch

Daniel Kroening†

*Computing Laboratory,
Oxford University,
OX1 3QD Oxford, UK.*

kroening@comlab.ox.ac.uk

Abstract

In this paper, we consider the problem of classifying Hamiltonian cycles in a binary hypercube. Previous work proposed a classification of these cycles using the edge representation, and presented it for dimension 4. We classify cycles in two further dimensions using a reduction to propositional SAT. Our proposed algorithm starts with an over-approximation of the set of equivalence classes, which is then refined using queries to a SAT-solver to remove spurious cycles. Our method performs up to three orders of magnitude faster than an enumeration with symmetry breaking in the 5-cube.

KEYWORDS: *bitonic sorting network, code equivalence, Gray code, Hamiltonian cycle, hypercube, SAT-solver, symmetry breaking*

Submitted November 2007; revised March 2008; published April 2008

1. Introduction

A *Hamiltonian cycle* is a closed loop through a graph that visits each node exactly once [1]. Various algorithms were invented to find Hamiltonian cycles, including construction algorithms (e.g., by extending a perfect matching of a hypercube [21]), neural approaches [3], using light rays to perform such computation [32], and others. Hamiltonian cycles and paths are of interest in many hypercube-like structures: in star graphs [22], in crossed hypercubes [39], in hypercubes with prescribed edges [17], and many more. In general, binary *hypercubes* are important in many areas of science, e.g., supercomputing [26], algorithmic biology [7], network protocols [6], and cryptography.

There is exactly one Hamiltonian cycle in the 2- and 3-cube up to automorphism of a hypercube [25]. The number of different Hamiltonian cycles was reported to be 1,344 and 906,545,790 for dimensions 4 and 5, respectively [25, 14]. The number of classes in higher dimensions is unknown, but bounds are well studied [15, 16, 30, 10, 34, 20].

P. P. Parkhomenko proposed a coarser classification of the Hamiltonian cycles using the *edge representation* [33]. An upper bound on the number of equivalence classes can be

* Supported by ETH Research Grant TH-19 06-3.

† Supported by an award from IBM Research.

Table 1. Hamiltonian Cycles

n	#cycles	#classes	
		by automorphisms	by edge weights
3	6	1	1 [33]
4	1,344 [15, 28]	9 [1]	4 [33]
5	906,545,790 [14]	237,675 [14]	28
6	$\leq 1.50378 \cdot 10^{30}$ [10]	unknown	550
7	$\leq 2.51511 \cdot 10^{67}$ [10]	unknown	$\leq$ 28972

computed as function of n (see Appendix A). Parkhomenko classified the 1,344 Hamiltonian cycles of the 4-cube into 4 equivalence classes. In this paper, we present the classification for two further dimensions (see Table 1, the numbers of equivalence classes in bold face are our findings).

The empowering technique behind our result is propositional satisfiability (SAT) solving. Given a Boolean formula in conjunctive normal form (CNF), a propositional SAT-solver determines if the formula is satisfiable, and if so, provides a satisfying assignment to the variables in the formula. Solvers for this problem have made tremendous progress over the past few years. This paper describes how to use a SAT-solver effectively to classify the Hamiltonian cycles in binary hypercubes. Our approach aims at only *classifying* them, as the problem of computing the number of cycles for a given class has no efficient algorithmic solution [33].

Our main contribution is *QUBS* (Queries for Upper Bound Strengthening), a classification method that computes a (conservative) set of candidates for equivalence classes. This set is then filtered using a propositional SAT-solver, and all spurious sequences are removed. The experimental results on the 5-cube show that this new method is three orders of magnitude faster than an enumeration with symmetry breaking. The experiments also show that the number of spurious equivalence classes is very small.

The paper is organized as follows: Section 2 introduces the necessary notation and definitions concerned with Hamiltonian cycles. In Section 3, we describe our encoding of Hamiltonian cycles as a propositional formula. In Section 4, we extend the SAT-encoding to classify Hamiltonian cycles. Section 4.2 describes an algorithm based on All-SAT that enumerates the equivalence classes by adding blocking clauses cycles equivalent to those already found. In Section 4.3, we improve the performance of this algorithm by breaking symmetries externally: we add constraints to the SAT-instance upfront, instead within the SAT engine. Section 4.4 provides the details of *QUBS*. It performs queries to improve the upper bound on the number of equivalence classes. The appendices present the calculation of these bounds for any given dimension, and the classes of Hamiltonian cycles in the 5- and 6-cube.

2. Hamiltonian Cycles

We begin by introducing definitions concerned with the Hamiltonian cycles in binary hypercubes.

2.1 Notation and Definitions

Definition 1 (Hypercube [14]). *Given a positive integer n , the binary n -cube Q_n is defined as the graph whose set of vertices is $\{0, 1\}^n$ and whose set of edges is formed by the pairs of vertices $A = (A[0], A[1], \dots, A[n-1])$ and $B = (B[0], B[1], \dots, B[n-1])$ that differ in just one coordinate $i \in \{0, 1, \dots, n-1\}$, i.e., $A[i] \neq B[i]$, and $A[j] = B[j]$ for all $j \neq i$.*

Definition 2 (Hamming distance). *The number of coordinates in which two vertices A and B differ is the Hamming distance: $d_H(A, B) := |\{i \mid A[i] \neq B[i]\}|$.*

Definition 3 (Hamiltonian cycle [14]). *A Hamiltonian cycle (or H-cycle) of an n -cube is a cycle of length 2^n that does not visit any vertex twice.*

Definition 4 (Coordinate sequence [23]). *The coordinate sequence of a cycle is a listing of the coordinates that change as this cycle is traversed. Thus, for the cycle $I_0, I_1, \dots, I_{k-1}$ of length k in an n -cube, the coordinate sequence contains k elements within the range $\{0, \dots, n-1\}$, where the i -th element indicates the index of the flipped bit while traversing from node I_i to node $I_{i+1 \bmod k}$.*

For instance, the H-cycle presented in Figure 1 traverses the nodes in the following order:

$$\begin{aligned} &(0000), (0001), (0011), (0010), (0110), (0111), (1111), (1011), \\ &(1001), (1000), (1010), (1110), (1100), (1101), (0101), (0100), \end{aligned} \quad (1)$$

and thus, its coordinate sequence is

$$(0 \ 1 \ 0 \ 2 \ 0 \ 3 \ 2 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 3 \ 0 \ 2) . \quad (2)$$

2.2 Equivalence of H-Cycles

Our classification of H-cycles follows the proposal of Parkhomenko [33], but we formulate it more concisely in terms of *change numbers* and *change sequences*.

Definition 5 (Change number [23]). *The change number c_j of the j^{th} coordinate is the number of times this coordinate appears in the coordinate sequence.*

Definition 6 (Change sequence [38]). *The change sequence is a vector $(c_{n-1}, \dots, c_0)$ of change numbers over all coordinates.¹*

Thus, the H-cycle of Eq. (1) has the change sequence $(2, 4, 4, 6)$.

We define the equivalence relation for H-cycles according to the one in [33].

1. This definition of *change sequence* is not to be confused with the definition of a *change-number sequence* [14], which is a coordinate sequence in our nomenclature.

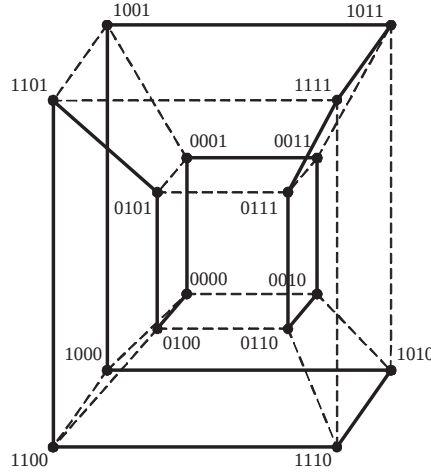


Figure 1. An H-cycle in the 4-cube (the solid lines are edges of the H-cycle)

Definition 7 (Equivalence of Hamiltonian Cycles). *Two H-cycles are equivalent if the change sequence for the first cycle can be obtained from the change sequence of the second cycle by a permutation of its change numbers.*

This equivalence relation is *coarser* than the one defined by automorphisms of a hypercube. In other words, if we choose two H-cycles, which are the same up to automorphisms of an n -cube, they fall in the same equivalence class with respect to Def. 7, but the inverse does not necessarily hold.

For instance, consider the H-cycle with the following coordinate sequence:

$$(3 \ 2 \ 1 \ 0 \ 3 \ 1 \ 3 \ 2 \ 3 \ 1 \ 3 \ 0 \ 3 \ 2 \ 1 \ 2) . \quad (3)$$

The change sequence of this cycle is $(6, 4, 4, 2)$ and by Def. 7, this H-cycle is equivalent to the one in (2). Note that these two H-cycles are not equivalent by automorphisms of a 4-cube.

It was shown in [33] that there exists a single class of Hamiltonian cycles in Q_3 and four classes in Q_4 . In this paper, we extend this classification by two dimensions.

2.3 Properties of Change Sequences

Let c be the change sequence of an H-cycle $I_0, I_1, \dots, I_{N-1}$ in Q_n , where $N = 2^n$. Then, the following properties hold.

Property 1 (properties 2 and 3 of [33]). *Each change number c_j of c is an even number that lies in the range $\{2, \dots, \frac{N}{2}\}$.*

For every cycle on a hypercube, if we traverse some axis in one direction, we have to return to the same hyperplane eventually, thus the change numbers are even. As an H-cycle visits all nodes, it has to visit every hyperplane as well, thus the change numbers are at

least 2. The change number may not exceed $\frac{N}{2}$ simply because this is the number of edges in a hyperplane, and every edge may be traversed only once.

Since every H-cycle has N vertices and edges, we can conclude the following property.

Property 2 ([33]). *The change numbers of an H-cycle sum up to the number of vertices of an n -cube: $\sum_{j=0}^{n-1} c_j = N$.*

These properties of change sequences determine an upper bound on the number of equivalence classes. The calculation of the bound is given in Appendix A. We use them for the QUBS algorithm presented in Section 4.4.

3. The SAT Encoding

3.1 The Propositional Satisfiability Problem

We employ a propositional SAT-solver to search for H-cycles. Propositional satisfiability (SAT) solvers have made tremendous progress in the past few years, and are able to solve many problems of practical interest with hundreds of thousands of variables. This progress was mainly driven by annual competitions, held in conjunction with the SAT conference series. All competitive solvers are based on the Davis-Putnam-Loveland-Logemann (DPLL) framework, which was introduced in the early sixties [12, 13]. The introduction of CHAFF in 2001 [31] marked a breakthrough in performance that led to renewed interest in the field. The authors of CHAFF proposed the idea of *conflict-driven non-chronological* backtracking coupled with the first conflict-driven decision heuristic.

Let V be a set of Boolean variables, i.e., $V = \{x_0, x_1, \dots, x_{m-1}\}$. A *literal* is a variable x_i or its negation $\neg x_i$. Let ϕ be a propositional formula over the variables in V . The propositional satisfiability problem [24] is to determine whether there exists an assignment of truth values to the variables in V such that the formula ϕ evaluates to *true*.

Most SAT-solvers accept propositional formulae in *Conjunctive Normal Form* (CNF). A formula in CNF is given as a set of clauses, where each clause is a disjunction of literals. Formulae that are not in CNF need to be translated. Tseitin introduced a linear-time translation of arbitrary Boolean formulae into CNF that preserves satisfiability [37]. We use Tseitin's transformation in our encodings whenever necessary, and present our constraints as Boolean formulae.

For our experiments, we use the MiniSat, written by Eén and Sörensson [19]. MiniSat provides interfaces for incremental solving and All-SAT. The current version uses preprocessing techniques from QBF-Solvers [18]. The rest of this section describes the encoding of the search for an H-cycle as an instance of the propositional SAT problem.

3.2 Propositional Encoding of H-Cycles

Our goal is a formula whose satisfying assignments correspond to the coordinates of the nodes forming an H-cycle in the n -cube. For this purpose, we define $n \cdot N$ Boolean variables denoted by $I_i[j]$, where $i \in \{0, \dots, N-1\}$ and $j \in \{0, \dots, n-1\}$. The Boolean vector I_i denotes the coordinates of node number i of the H-cycle, where $I_i[0]$ corresponds to the right-most bit of the coordinates of node I_i .

In what follows, we define constraints over these variables using propositional formulae. The constraints evaluate to true if and only if the $I_i[j]$ correspond to some H-cycle

$I_0, I_1, \dots, I_{N-1}$. We then apply a SAT-solver to obtain a solution. Trivially, the coordinates of the H-cycle can be extracted from any satisfying assignment provided by the solver.

The sequence of nodes $I_0, I_1, \dots, I_{N-1}$ ought to meet the following requirements in order to represent an H-cycle (the propositional encoding of the Hamming distance between two nodes is described in the next subsection):

1. To form a cycle in an undirected n -cube, the neighboring nodes must be adjacent in the hypercube. The adjacency is expressed using the Hamming distance:

$$\phi^{cycle} := \bigwedge_{i=0}^{N-1} (d_H(I_i, I_{i+1 \bmod N}) = 1) . \quad (4)$$

2. For a cycle to be *Hamiltonian*, no node may appear twice, so all binary words must be distinct. That is, the Hamming distance between any two nodes must be at least one:².

$$\phi^{distinct} := \bigwedge_{i=0}^{N-3} \bigwedge_{j=i+2}^{N-1} (d_H(I_i, I_j) \geq 1) . \quad (5)$$

As an optimization, we encode this constraint only for the nodes of same parity, i.e., when $j \bmod 2 = i \bmod 2$, because otherwise nodes are distinct by 2-colorability of a hypercube (Eq. (4) assures the color alternation).

The propositional formula

$$\phi^{HC} := \phi^{cycle} \wedge \phi^{distinct} \quad (6)$$

encodes the H-cycle in the n -cube. A satisfying assignment to ϕ^{HC} contains the coordinates of some H-cycle. We now explain how to encode the Hamming distance $d_H(A, B)$ for a pair of nodes A and B .

3.3 Encoding the Hamming distance

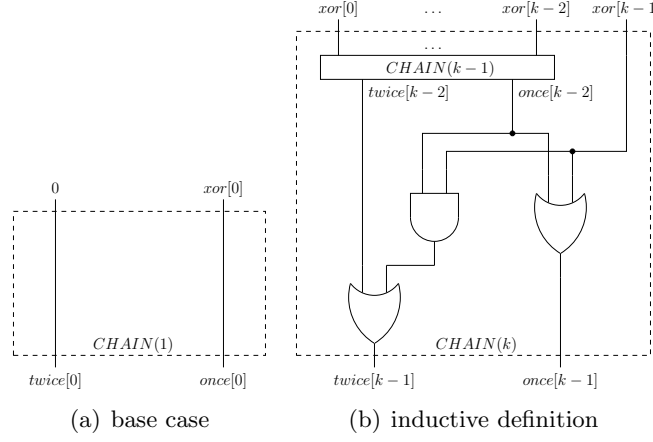
Consider two nodes A and B of the n -cube for which the Hamming distance has to be computed.

Let $xor^{A,B}$ denote the bitwise XOR of the nodes A and B , i.e., we introduce n new Boolean variables $xor^{A,B}[j]$ indicating whether the vectors A and B differ in the j -th bit:

$$\bigwedge_{j=0}^{n-1} (xor^{A,B}[j] \longleftrightarrow A[j] \oplus B[j]) . \quad (7)$$

The next step would be to encode the sum of $xor^{A,B}[j]$ for all $j \in \{0, \dots, n-1\}$. However, observe that computing the precise value of the Hamming distance is not actually needed to achieve our goal. Instead, it is sufficient to check whether the Hamming distance is equal to one for formula (4) or greater or equal to one for (5). We therefore introduce Boolean variables that indicate if the Hamming distance is at least one or at least two, respectively. Let the former be called *once* and the latter *twice*. Subsequently, given two nodes A and B , we are able to encode

2. As an optimization, this constraint is not generated for two neighbors in the sequence, which have distance one anyways.


Figure 2. Chain encoding

- the fact that $d_H(A, B)$ is exactly one as

$$once^{A,B} \wedge \neg twice^{A,B} , \quad (8)$$

- and the fact that $d_H(A, B) \geq 1$ as

$$once^{A,B} . \quad (9)$$

We experimented with different ways of encoding *once* and *twice*. The best encoding we found uses a chain-construction. We introduce n variables $once^{A,B}[j]$ for $j \in \{0, \dots, n-1\}$ indicating that A and B differ in *at least one* position *within* the range $\{0, \dots, j\}$. The $twice^{A,B}[j]$ variables are defined in the similar way to indicate two or more bit-flips.

The definitions of the variables *once* and *twice* are inductive (see also Figure 2):

$$(once^{A,B}[0] := xor^{A,B}[0]) , \quad (10)$$

$$\bigwedge_{j=1}^{n-1} (once^{A,B}[j] \longleftrightarrow once^{A,B}[j-1] \vee xor^{A,B}[j]) , \quad (11)$$

$$(twice^{A,B}[0] := \text{false}) , \quad (12)$$

$$\bigwedge_{j=1}^{n-1} (twice^{A,B}[j] \longleftrightarrow twice^{A,B}[j-1] \vee once^{A,B}[j-1] \wedge xor^{A,B}[j]) . \quad (13)$$

An alternative encoding is to make use of a balanced tree to compute $d_H \geq 1$, together with NAND operations for all pairs of XORs to compute $d_H \leq 1$:

$$d_H(A, B) \geq 1 \longleftrightarrow \bigvee_{j=0}^{n-1} xor^{A,B}[j] , \quad (14)$$

$$d_H(A, B) \leq 1 \longleftrightarrow \bigwedge_{j=0}^{n-2} \bigwedge_{k=j+1}^{n-1} (\neg xor^{A,B}[j] \vee \neg xor^{A,B}[k]) . \quad (15)$$

This *tree encoding* results in a smaller depth and size of the formula. Nevertheless, we have observed that the SAT-solver performs better on the “once-twice” chain encoding

Table 2. Different encodings of an H-cycle

Encoding		n	#vars		#clauses			Time to solve (s)
ϕ_{cycle}	$\phi_{distinct}$		#xors	total	ϕ_{cycle}	$\phi_{distinct}$	total	
tree		5	1360	2608	1376	7925	9301	0.10
tree	once-twice			4769		13924	15300	0.05
once-twice	tree			2896	1888	7925	9813	0.18
once-twice chain				5057		13924	15812	0.06
tree		6	6336	12000	3520	39686	43206	7.24
tree	once-twice			22912		70438	73958	1.83
once-twice	tree			12704	4608	39686	44294	23.87
once-twice				23616		70438	75046	1.54

(see Table 2). On Q_6 , the run-time using the chain encoding is 78% lower than using the tree encoding.³ This improvement is a bit surprising, since the “once-twice” encoding constitutes almost twice as many clauses as the tree encoding. One possible reason for the improvement is the better propagation of decisions by the SAT-solver between parts of the encoding of ϕ_{cycle} : in case of the tree encoding, the subformulae (14) and (15) are more isolated.

4. Classification

In this section, we present the classification of Hamiltonian cycles with respect to permutations of a change sequence. We have reported the classification of *induced cycles* in binary hypercubes in earlier work [40]. That classification was performed by enumerating *all cycles* using an *All-SAT* solver (i.e., a SAT-solver that computes all satisfying assignments of a propositional formula) and a subsequent classification. The evident disadvantage of this approach is that it relies on the set of all cycles being enumerable. However, a 5-cube already contains over a billion Hamiltonian cycles (see Table 1), and the number of cycles in Q_6 is not reported yet.

Instead of enumerating all H-cycles and subsequently classifying them, we use the SAT-solver to perform the classification. To that end, we need to encode the change sequence of an H-cycle.

4.1 Encoding the Change Sequence

We compute the change number c_j of the j -th coordinate as the sum of the *xor* variables over this coordinate:

$$c_j = \sum_{i=0}^{N-1} \text{xor}^{I_i, I_{i+1 \bmod N}}[j]. \quad (16)$$

3. All experiments in this paper were performed on a PC with 16 GB of memory and a 3 GHz processor.

A straight-forward encoding uses N adders to compute this sum. Initial experimental results, however, indicate that there is a superior encoding: using a *sorting network* we can obtain a *unary* representation of the value of c_j (which suffices for our method). The sorting network receives *xor* bits from Eq. (16) as input and produces a sequence of zeros followed by ones as output (see [27] for a survey).

Let variables $\sigma_j[i]$ with $i \in \{0, \dots, N-1\}$ be the unary representation of c_j , i.e., $c_j = \sum_{i=0}^{N-1} \sigma_j[i]$. We claim the following:

$$(\sigma_j[i] \longrightarrow c_j \geq N-i) \wedge (\neg \sigma_j[i] \longrightarrow c_j \leq N-i),$$

for all $j = \{0, \dots, n-1\}$ and $i = \{0, \dots, N-1\}$.

The straight-forward encoding of a sorting network with k inputs has k^2 comparators and depth k^2 . Sorting networks can be constructed with $O(k \log_2 k)$ comparators [2], but this bound hides a large constant. A *bitonic sorting network*, as proposed by Batcher [5], has $O(k \log_2^2 k)$ comparators and depth $O(\log_2 k)$. We have applied bitonic sorting networks successfully in the past in SAT-encodings in a different context [11]. We omit details of the bitonic sorting networks encoding for the sake of brevity, but the number of additional variables is twice the number of comparators, as we need two variables to encode every comparator.

Compared to the quadratic sorting network, the encoding using the bitonic sorting network results in a run-time reduction of about 50% on Q_6 , and thus, all results reported are obtained using the bitonic sorting networks.

4.2 Direct Enumeration with Internal Symmetry Breaking

The classification algorithm is embedded *into* a SAT-solver. More precisely, we modify the algorithm of [41] to compute the number of equivalence classes directly. The only difference from the approach in [41] is the considered type of symmetry. In this approach, we solve an instance of an All-SAT problem, where variables that encode change numbers are set to be *important*. In other words, the SAT-solver enumerates *all* solutions to a propositional formula, for which valuations of $\sigma_j[i]$ differ.

To achieve this, we add a *blocking clause* for each H-cycle equivalent to the found one according to Def. 7. There are up to $n!$ types of H-cycles that are equivalent to the found one—the number of permutations of n change numbers in the change sequence. The search is repeated until no satisfying assignment can be found. Thus, every satisfying assignment returned corresponds to a representative of a different equivalence class.

The classification for Q_4 was obtained in less than one second, with results conforming to the findings of Parkhomenko in [33]. In dimension 5, the classification took 31 days (see Table 3).

4.3 Direct Enumeration with External Symmetry Breaking

Clearly, for our classification according to Def. 7, change numbers of an H-cycle are interchangeable. In fact, one could break these symmetries right in the SAT instance, instead of adding $n!$ blocking clauses as in the previous algorithm. In what follows, we transform a change sequence to *standard representation* [35]. We do that by encoding a *total order* on

Table 3. Classification of Hamiltonian Cycles

n	#classes	Direct enumeration		QUBS		
		internal symmetry breaking	external symmetry breaking	Time (s)	#candidates	Avg. time (s)
		Time (s)	Time (s)			
4	4	<1	<1	<1	4	<1
5	28	2,680,000	162,272	106	30	3.5
6	550	-	-	190,852	596	320
7	unknown	-	-	-	30,249	-

the change numbers, i.e. $c_j \leq c_k$ for all $j \leq k$, using the following clauses:

$$\bigwedge_{j=1}^{n-1} \bigwedge_{i=0}^{N-1} (\sigma_{j-1}[i] \longrightarrow \sigma_j[i]) . \quad (17)$$

The only disadvantage of the symmetry breaking directly in a SAT instance is that we add new $n \cdot N$ clauses. On the other hand, for every obtained cycle we add only one blocking clause instead of $n!$. Moreover, one can use an unmodified all-solutions SAT-solver to obtain the classification.

The classification for Q_5 using the external symmetry breaking approach was obtained within 2 days (instead of 31 days in case of internal symmetry breaking). Since this approach was still not promising for Q_6 , we attempted to improve the performance by shifting some complexity of the problem away from the SAT-solver.

4.4 Queries for Upper Bound Strengthening (QUBS)

The performance bottleneck for the enumeration is in proving that all classes have been computed. This corresponds to establishing a conflict in the All-SAT procedure. For instance, in the classification for the 5-cube using internal and external symmetry breaking, the SAT-solver spends 99.5% and 79% respectively of the computation time in establishing a conflict. The larger the search space, the greater the effort required to reach a conflict.

Unlike in the previous methods, where the set of equivalence classes EC is constructed iteratively, we propose to consider a conservative over-approximation of EC . We call this set the set of *equivalence class candidates* (ECC). The candidates are defined using the properties 1 and 2.⁴ It was already pointed out in [33] that not all change sequences obeying these properties correspond to feasible Hamiltonian cycles on an n -cube. We call such sequences *spurious*.

In order to eliminate the spurious sequences, we perform $|ECC|$ queries, each of which requires a run of the SAT-solver. For every change sequence (candidate), we generate an instance where the values of $\sigma_j[i]$ are fixed appropriately. If the instance is satisfiable, the H-cycle exists. Clearly, the number of satisfiable instances is the number of equivalence

4. The derivation of the size of ECC is shown in Appendix A.

classes of H-cycles. If it is unsatisfiable, the candidate is spurious, and we can strengthen the upper bound.⁵ For instance, in Q_5 , the Hamiltonian cycles with change sequences (2, 2, 2, 10, 16) and (2, 2, 2, 12, 14) are spurious.

Our experiments on Q_5 show that there is a dramatic improvement in performance: the speedup is around three orders of magnitude. Also, only 7% of the candidates are spurious, thus, little work is wasted. We have therefore conducted the classification for the 6-cube using QUBS (see Table 3). The number of candidates is 596, out of which 550 cycles are feasible. Thus, the number of spurious cycles is less than 8% on Q_6 . The overhead caused by the spurious cycles is negligible—the unsatisfiable instances took about 0.46% of the total run-time.

The number of candidates in Q_7 is 30,249. The time required for solving the SAT instances generated from queries varies significantly between candidates: some are easily solved, whereas others time out. In the future, we expect to obtain at least a strong upper bound for Q_7 .

In summary, this approach reduces the size of the search space significantly. Moreover, this method is fully parallelizable. The only disadvantage is that this approach requires $|ECC|$ calls to the SAT-solver, and this number is at least the number of equivalence classes.

4.5 Discussion and Future Work

Taking a closer look at spurious candidates, we can deduce the following:

Observation 1. *In hypercubes up to dimension 7, H-cycles with change sequences lexicographically less than that of a Binary Reflected Gray Code (BRGC) [35] do not exist.*

Indeed, two spurious candidates in the 5-cube have change sequences (2, 2, 2, 10, 16) and (2, 2, 2, 12, 14), and are less than $c^{BRGC} = (2, 2, 4, 8, 16)$. We have conducted a specialized SAT-based search to verify this observation in the 7-cube. This search was targeted not on individual change sequences, but candidates with a given prefix of the change sequences, e.g., (2, 2, 4, 8, 8, *, *), allowing the SAT-solver to choose the last two change numbers. The algorithm terminated within 4 days with *UNSAT*, and confirmed that 1277 candidates with change sequences less than that of the BRGC are spurious (thus, allowing us to tighten the bound of the number of classes, see Table 1). Note that observation 1 is *not sufficient* to filter out all spurious cycles, e.g., candidates with prefix (2, 4, 4, 4, *, *) in the 6-cube are spurious even though this sequence is greater than that of the BRGC.

The authors are not familiar with any related work stating or proving⁶ observation 1. The following may serve as an intuition why the aforementioned candidates are spurious: they are rather *unbalanced*, i.e., their change numbers differ a lot. In fact, Hamiltonian cycles with *balanced* change sequences are well studied and are of interest in design and testing of electrical circuits [8, 38, 29]. The search by QUBS can be restricted to balanced Hamiltonian cycles, as QUBS allows us to explicitly specify a change sequence. We were able to find the cycles with the change sequences mentioned in [29] within 3 hours.

5. This approach is inspired by [9], and we therefore use the terminology used in that reference.

6. V. N. Potapov has a proof of a stronger criterion, which is not published yet.

5. Conclusion

In this paper, we presented the classification of the Hamiltonian cycles in the 5- and 6-cubes with respect to symmetries of the change sequence, extending the previous results by two dimensions. Our main contribution is a classification method that computes a set of candidates for equivalence classes. Spurious sequences are then filtered from the set using a propositional SAT-solver. In the 5-cube, the new method is three orders of magnitudes faster than an enumeration with symmetry breaking.

In the future, we plan to extend the presented classification of Hamiltonian cycles by one dimension and to enumerate classes with respect to automorphisms.

Acknowledgments

We acknowledge the helpful discussions with Christoph M. Wintersteiger and Gennady Shmonin regarding partitioning theory. We thank Thomas Wahl, Angelo Brillout, Roman Mitin, Mitra Purandare, Vijay V. D'silva and the anonymous reviewers for improving this script.

References

- [1] H. L. Abbott. Hamiltonian circuits and paths on the n -cube. *Canad. Math. Bull.*, 9:557–562, 1966.
- [2] M. Ajtai, J. Komlós, and E. Szemerédi. An $O(n \log n)$ sorting network. In *STOC '83: Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 1–9. ACM, 1983.
- [3] I. K. Altinel, N. Aras, and B. J. Oommen. Fast, efficient and accurate solutions to the Hamiltonian path problem using neural approaches. *Comput. Oper. Res.*, 27(5):461–494, April 2000.
- [4] G. E. Andrews. *The theory of partitions*. Addison-Wesley, 1976.
- [5] K. E. Batchier. Sorting networks and their application. In *AFIPS Conf. Proc.*, volume 32, pages 307–314. AFIPS Press, 1968.
- [6] K. Becker and U. Wille. Communication complexity of group key distribution. In *Procs. of the 5th ACM Conference on Computer and Communications Security*, pages 1–6. ACM, 1998.
- [7] T. Y. Berger-Wolf, C. Moore, and J. Saia. A computational approach to animal breeding. *J Theor Biol*, 244(3):433–439, February 2007.
- [8] G. S. Bhat and C. D. Savage. Balanced Gray codes. *The Electronic Journal of Combinatorics*, 3(R25):1–11, 1996.
- [9] A. R. Bradley and Z. Manna. Checking safety by inductive generalization of counterexamples to induction. In *Formal Methods in Computer-Aided Design (FMCAD)*, pages 173–180. IEEE Computer Society, 2007.
- [10] L. H. Clark. A new upper bound for the number of Hamiltonian cycles in the n -cube. *J. Combin. Inform. System Sci.*, 25:35–37, 2000.
- [11] E. Clarke, D. Kroening, J. Ouaknine, and O. Strichman. Computational challenges in bounded model checking. *Software Tools for Technology Transfer (STTT)*, 7(2):174–183, April 2005.
- [12] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.
- [13] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.
- [14] I. J. Dejter and A. A. Delgado. Classes of Hamilton cycles in the 5-cube. *J. Combinat. Math, Combinat. Comput*, 61:81–95, 2007.
- [15] E. Dixon and S. Goodman. On the number of Hamiltonian circuits in the n -cube. In *Proc. of the Amer. Math. Soc.*, volume 50, pages 500–504, 1975.

- [16] R. J. Douglas. Bounds on the number of Hamiltonian circuits in the n -cube. *Discrete Math.*, 17:143–146, 1977.
- [17] T. Dvoraák. Hamiltonian cycles with prescribed edges in hypercubes. *SIAM J. Discret. Math.*, 19(1):135–144, 2005.
- [18] N. Eén and A. Biere. Effective preprocessing in sat through variable and clause elimination. In *Theory and Applications of Satisfiability Testing (SAT)*, volume 3569 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2005.
- [19] N. Eén and N. Sörensson. An extensible SAT-solver. In *Theory and Applications of Satisfiability Testing*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2004.
- [20] T. Feder and C. Subi. Nearly tight bounds on the number of Hamiltonian circuits of the hypercube and generalizations. Technical Report 63, Electronic Colloquium on Computational Complexity, 2007.
- [21] J. Fink. Perfect matchings extend to Hamilton cycles in hypercubes. *J. Comb. Theory*, 97(6):1074–1076, November 2007.
- [22] J.-S. Fu. Conditional fault-tolerant hamiltonicity of star graphs. *Parallel Comput.*, 33(7-8):488–496, 2007.
- [23] L. Glass. Combinatorial aspects of dynamics in biological systems. In U. Landman, editor, *Statistical mechanics and statistical methods in theory and applications*, pages 585–611. Plenum, 1977.
- [24] J. Gu, P. Purdom, J. Franco, and B. Wah. Algorithms for the satisfiability (SAT) problem: a survey. *DIMACS, Amer. Math. Soc.*, pages 19–152, 1997.
- [25] F. Harary, J. P. Hayes, and H.-J. Wu. A survey of the theory of hypercube graphs. *Comput. Math. Appl.*, 15(4):277–289, 1988.
- [26] J. P. Hayes and T. Mudge. Hypercube supercomputers. In *Procs. of the IEEE*, volume 77, pages 1829–1841, 1989.
- [27] D. E. Knuth. *The Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley, 1973.
- [28] G. Kreweras. Some remarks about Hamiltonian circuits and cycles on hypercubes. *Bull. Inst. Comb. Appl.*, 12:19–22, 1994.
- [29] X. Liu and G. F. Schrack. A heuristic approach for constructing symmetric Gray codes. *Appl. Math. Comput.*, 155(1):55–63, July 2004.
- [30] M. Mollard. New bounds for the number of Hamiltonian cycles of an n -cube. *European J. of Combin.*, 9(1):49–52, 1988.

- [31] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: engineering an efficient SAT solver. In *Procs of the 38th Design Automation Conference (DAC)*, pages 530–535. ACM, 2001.
- [32] M. Oltean. Solving the Hamiltonian path problem with a light-based computer. *Natural Computing: an international journal*, 7(1):57–70, 2008.
- [33] P. P. Parkhomenko. Classification of the Hamiltonian cycles in binary hypercubes. *Autom. Remote Control*, 62(6):978–991, 2001.
- [34] A. L. Perezhogin and V. N. Potapov. On the number of Hamiltonian cycles in a Boolean cube. *Diskretn. Anal. Issled. Oper.*, 8(2):52–62, 2001.
- [35] C. Savage. A survey of combinatorial gray codes. *SIAM Rev.*, 39(4):605–629, 1997.
- [36] J. A. Todd. A table of partitions. In *Proc. London Math. Soc.*, volume 48, pages 229–242, 1943.
- [37] G. S. Tseitin. On the complexity of derivation in propositional calculus. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning 2: Classical Papers on Computational Logic 1967-1970*, pages 466–483. Springer, 1983.
- [38] V. E. Vickers and J. Silverman. A technique for generating specialized Gray codes. *IEEE Trans. Comput.*, 29(4):329–331, 1980.
- [39] D. Wang. An algorithm to embed Hamiltonian cycles in crossed cubes. In *Procs. of the international symposium on Parallel Computing in Electrical Engineering*, pages 49–54. IEEE Computer Society, 2006.
- [40] I. Zinovik, D. Kroening, and Y. Chebiryak. An algebraic algorithm for the identification of Glass networks with periodic orbits along cyclic attractors. In Hirokazu Anai, Katsuhisa Horimoto, and Temur Kutsia, editors, *AB*, volume 4545 of *Lecture Notes in Computer Science*, pages 140–154. Springer, 2007.
- [41] I. Zinovik, D. Kroening, and Y. Chebiryak. Computing binary combinatorial Gray codes via exhaustive search with SAT-solvers. *IEEE Transactions on Information Theory (IT)*, 54(4):1819–1823, April 2008.

Appendix A. The Number of Candidates

In this section, we give details on the upper bound for the number of equivalence classes used by QUBS.

Definition 8 ([4]). *A partition of a positive integer n is a finite non-increasing sequence of positive integers $\lambda_1, \lambda_2, \dots, \lambda_r$ such that $\sum_{i=1}^r \lambda_i = n$. The λ_i are called the parts of the partition.*

Let $p(N, \leq M, n)$ denote the number of partitions of n into at most M parts, each $\leq N$, $p(N, = M, n)$ for exactly M parts and $p_{\text{even}}(\dots)$ denote partitioning into even parts.

It follows from properties 1 and 2 that the number of candidates for equivalence classes equals the number of partitions of 2^n into n even parts, each of which is at least 2 and at most 2^{n-1} : $p_{\text{even}}(2^{n-1}, =n, 2^n)$. It can be seen that the latter is the same as the number of partitions of 2^{n-1} into n parts, each $\leq 2^{n-2}$:

$$\begin{aligned} p_{\text{even}}(2^{n-1}, =n, 2^n) &= \\ &= p(2^{n-2}, =n, 2^{n-1}) \\ &= p(2^{n-2}, \leq n, 2^{n-1}) - p(2^{n-1}, \leq (n-1), 2^{n-1}) \end{aligned} \tag{18}$$

$$= p(2^{n-2} - 1, \leq n, 2^{n-1} - n) \tag{19}$$

$$\leq p(2^{n-1} - n, \leq n, 2^{n-1} - n) = p(2^{n-1}, =n, 2^{n-1}) \tag{20}$$

The simplification (18) follows from the inclusion-exclusion principle, and Eq. (19) is obtained using formula (3.2.6) of [4]. Eq. (20) gives us an upper bound of 5, 37, 709, and 34082 candidates for dimensions from 4 to 7, respectively [36]. Eq. (19) evaluates to 4, 30, 596, and 30249 candidates, respectively.

Appendix B. Equivalence Classes in the 5-cube.

Table 4. Equivalence Classes of Hamiltonian Cycles in dimension 5

Coordinate sequence	Change sequence
0 2 0 1 0 3 0 1 0 2 0 1 0 4 0 1 0 2 0 1 0 3 0 1 0 2 0 1 0 4 0 1	(2,2,4,8,16)
0 1 0 2 0 1 0 3 1 0 1 2 0 1 0 4 0 1 0 2 1 0 1 3 0 1 0 2 0 1 0 4	(2,2,4,10,14)
2 0 1 0 4 1 0 1 2 1 0 1 3 0 1 0 2 0 1 0 4 0 1 0 2 1 0 1 3 1 0 1	(2,2,4,12,12)
0 1 0 4 0 1 0 2 0 1 0 3 0 2 0 1 0 2 0 4 0 2 0 1 0 2 0 3 0 1 0 2	(2,2,6,6,16)
4 0 1 0 2 0 1 0 3 0 1 0 2 0 1 0 4 2 0 1 0 2 0 1 3 1 0 2 0 1 0 2	(2,2,6,8,14)
1 3 0 2 0 1 0 2 0 4 2 1 0 1 2 1 0 3 0 1 0 2 0 1 0 4 1 0 1 2 1 0	(2,2,6,10,12)
0 2 0 1 0 2 0 3 0 1 2 1 0 1 2 4 2 0 1 0 2 0 1 3 1 0 2 0 1 0 2 4	(2,2,8,8,12)
2 0 1 0 4 1 2 1 0 1 2 1 3 0 2 1 2 0 2 1 4 0 2 0 1 0 2 0 3 1 0 1	(2,2,8,10,10)
0 2 0 1 0 4 0 1 0 3 0 1 0 2 0 1 0 3 0 1 0 4 0 3 0 2 0 1 0 2 0 3	(2,4,4,6,16)
1 0 1 4 0 1 0 3 0 1 0 2 0 3 0 1 0 3 0 4 0 2 0 1 0 2 0 3 1 0 1 2	(2,4,4,8,14)
2 0 3 0 1 0 3 0 2 0 1 3 1 0 1 4 1 0 1 2 1 0 1 3 0 1 0 2 0 1 0 4	(2,4,4,10,12)
1 0 2 0 3 1 0 1 2 0 1 0 4 3 0 3 2 0 1 0 2 0 3 0 2 0 1 0 4 0 2 0	(2,4,6,6,14)
0 3 1 2 1 0 2 1 2 3 2 0 4 0 1 0 3 0 1 0 2 0 1 0 3 0 1 0 4 1 0 2	(2,4,6,8,12)
2 1 0 1 2 3 4 0 1 0 2 1 0 1 3 1 0 1 2 1 0 1 4 0 3 0 2 0 1 0 2 3	(2,4,6,10,10)
3 0 1 4 2 1 0 1 2 1 0 3 2 1 2 0 1 2 1 4 0 2 0 3 2 0 1 0 3 0 2 0	(2,4,8,8,10)
3 0 3 1 0 3 0 4 1 2 0 2 3 0 2 0 1 0 2 0 3 2 0 4 1 0 1 3 0 1 0 2	(2,6,6,6,12)
2 4 1 2 3 0 2 0 1 0 2 0 3 0 1 2 1 4 1 0 3 0 1 0 2 1 3 0 3 1 3 0	(2,6,6,8,10)
3 1 0 2 3 2 0 2 3 1 3 0 3 2 4 2 0 1 0 2 0 1 3 1 2 1 0 1 2 1 4 0	(2,6,8,8,8)
0 4 0 3 0 1 0 2 0 1 0 4 0 3 0 4 0 1 0 2 0 4 0 2 0 3 0 2 0 1 0 3	(4,4,4,4,16)
2 0 1 0 3 0 1 0 4 0 1 0 2 0 4 0 1 0 3 1 0 4 0 2 0 1 0 2 3 0 3 4	(4,4,4,6,14)
1 4 3 1 0 1 3 4 2 0 1 0 3 2 0 1 0 2 0 1 4 0 1 0 3 0 1 0 2 0 4 0	(4,4,4,8,12)
0 4 0 1 0 3 1 4 1 0 2 1 0 1 3 2 0 2 1 0 4 1 0 1 3 0 1 0 2 1 4 3	(4,4,4,10,10)
0 4 0 2 1 3 0 2 1 0 2 0 4 0 1 0 3 0 2 0 3 1 0 1 4 3 0 2 1 4 0 2	(4,4,6,6,12)
2 0 2 3 0 2 0 1 0 4 0 3 1 0 1 4 0 2 3 0 4 1 2 0 2 1 3 1 0 1 4 1	(4,4,6,8,10)
2 0 1 0 2 4 2 1 0 1 3 0 1 2 3 0 1 4 3 2 0 1 2 1 4 0 1 2 4 2 3 0	(4,4,8,8,8)
3 4 1 3 0 3 2 0 4 2 3 0 2 1 0 1 4 0 1 0 2 0 3 2 0 4 0 1 2 1 3 0	(4,6,6,6,10)
3 2 1 2 3 4 1 0 3 0 1 0 4 1 2 1 3 0 4 3 0 1 0 3 0 1 2 4 1 2 0 2	(4,6,6,8,8)
0 2 3 1 3 2 3 1 4 1 0 1 2 0 1 0 4 3 4 0 2 0 4 3 1 2 3 4 0 2 0 4	(6,6,6,6,8)

Appendix C. Equivalence Classes in the 6-cube.

Table 5. Spurious Candidates for Hamiltonian Cycles in dimension 6

Change sequence
(2, 2, 2, *, *, *)
(2, 2, 4, 4, *, *)
(2, 2, 4, 6, *, *)
(2, 4, 4, 4, *, *)

Here stars denote that a change number can take arbitrary value (not contradicting properties 1 and 2). For any such valuations a change sequence of the cycle is spurious.