MATHEMATICS OF COMPUTATION
Volume 83, Number 286, March 2014, Pages 979–995
S 0025-5718(2013)02741-X
Article electronically published on July 15, 2013

COUNTING HAMILTONIAN CYCLES IN BIPARTITE GRAPHS

HARRI HAANPÄÄ AND PATRIC R. J. ÖSTERGÅRD

Abstract. A method for counting Hamiltonian cycles in bipartite graphs is developed with the main focus on the long-standing open case of the 6-cube. Dynamic programming as well as utilization of the automorphism group of the graph are central ingredients of the method. It is further shown how the number of equivalence classes of Hamiltonian cycles can be obtained via a classification of Hamiltonian cycles with nontrivial automorphisms. It turns out that the 6-cube has 35 838 213 722 570 883 870 720 Hamiltonian cycles and 777 739 016 577 752 714 equivalence classes of Hamiltonian cycles. The old result on the number of knight's tours on a 8×8 chessboard is confirmed.

1. Introduction

We consider connected undirected simple graphs, and use the notation $\Gamma = (V, \mathcal{E})$ for such a graph and $\mathrm{Aut}(\Gamma)$ for its automorphism group. A *Hamiltonian cycle* in Γ is a cycle that visits every vertex of V exactly once. A graph that has a Hamiltonian cycle is said to be *Hamiltonian*. The problems of finding necessary and sufficient conditions for graphs to be Hamiltonian are central in graph theory [33, Section 7.2]. Algorithmic issues related to Hamiltonian cycles are also of (theoretical and practical) importance. The problem of determining whether a graph is Hamiltonian is NP-complete [9, p. 199], and the problem of counting the Hamiltonian cycles in a graph is #P-complete [31].

Many deterministic and nondeterministic algorithms have been proposed for the problem of finding a Hamiltonian cycle in a graph [32]. Heuristic algorithms can be used to find one Hamiltonian cycle, but to prove that none exists or to find all Hamiltonian cycles, exhaustive methods are required. Backtrack search can be used to find all cycles. For proving nonexistence, linear programming has also been applied [18, p. 566].

A search for all Hamiltonian cycles gives the total number of such cycles. However, if one is just interested in this number, one may consider methods that do not construct all cycles explicitly. In fact, if the total number is immense, say greater than 10^{20}, it is not even computationally feasible to obtain explicit copies of all Hamiltonian cycles. Knuth [17, pp. 254–255] uses ZDD methods to count Hamiltonian cycles. The current work presents another method, which is based on dynamic programming. In dynamic programming, a problem is solved by first solving various subproblems, the solutions of which are combined to reach an overall solution. The new method is particularly elegant for bipartite graphs, so the consideration here is restricted to such graphs.

Received by the editor November 7, 2011 and, in revised form, March 12, 2012 and June 25, 2012.

2010 *Mathematics Subject Classification.* Primary 05C45, 05C85, 05C30, 05C38, 68R10.

Key words and phrases. Bipartite graph, enumeration, Hamiltonian cycle, hypercube.

The paper is organized as follows. A method for counting Hamiltonian cycles is developed in Section 2. In Section 3, the n-cube is used as an example, and the number of Hamiltonian cycles in the 6-cube is determined. The number of Hamiltonian cycles up to the symmetry of the 6-cube is further obtained using a combination of analytical and computational techniques. The classic example of knight's tours on the chessboard is considered in Section 4, and the paper is concluded in Section 5. It is assumed that the reader has a basic understanding of combinatorial algorithms [21].

2. Counting Hamiltonian cycles

2.1. **General approach.** The most straightforward exhaustive search method for Hamiltonian cycles is to pick one vertex—then we have a path of length 0—and add edges to the endpoints of the path to make it longer and longer. One may also have a collection of several paths as a partial solution; for the early history of such *multipath algorithms* and other related methods, see [2, Chapter 10], [27], and their references. These algorithms were later developed further by Kocay [19]; see also [20, Chapter 9]. Other proposed backtrack algorithms include [23, 28].

Given a partial Hamiltonian cycle as a collection of paths, the only information needed for completing a cycle is the set of pairs of endpoints of the paths and the set of vertices in the paths; the way the paths visit these vertices is irrelevant. This is well known; see [19, p. 175], where this fact is used for developing data structures for this kind of search. If the backtrack search is breadth-first rather than depth-first, then this fact can be used to combine partial solutions and thereby speed up the algorithm. (Depth-first search [2, 19] is useful in particular when one wants to find one Hamiltonian cycle fast.)

The general idea of gaining speed by solving a problem via a reduced number of subproblems fits the framework of *dynamic programming* [4, Chapter 15]. To be able to utilize this idea effectively, the approach for constructing partial solutions should be carefully developed. Obviously, the number of different (combined) partial solutions has to be within a range that can be saved and handled by the computers available.

From now on, we consider bipartite graphs only, and let $\Gamma = (V, \mathcal{E})$ be such a graph. The vertices of Γ can be partitioned into nonempty independent sets V_i, $0 \le i \le M$, such that every edge in $\mathcal{E}$ has its endpoints in consecutive sets V_k and V_{k+1} for some k. Note that a given graph may have different such partitions, with different numbers of independent sets (that is, $M + 1$). The choice of partitioning will have an impact on the performance of the algorithm to be developed.

The following observation can be made regarding a Hamiltonian cycle in a bipartite graph with the given vertex partitioning. The subgraph induced by the vertices in $\bigcup_{j=0}^{i} V_j$ for any given i contains a collection of paths that visit all those vertices and have their endpoints in V_i. We gather the endpoints of the paths of length at least 1 in S' and, to indicate which vertices are endpoints of the same path, as unordered pairs in $\mathcal{S}$. The vertices of the paths of length 0 are gathered in T. Thereby complete information about the endpoints in V_i is given by $(\mathcal{S}, T)$. The vertices in V_i not occurring in that structure are obviously internal vertices of paths (there is no need to state them explicitly since they can be deduced). This makes it possible to group solutions by the structure of their vertices in V_i as described above.

The discussed observation and data structures naturally lead to the following approach. We iteratively consider the subgraphs induced by $\bigcup_{j=0}^{i} V_j$ for $i = 0, 1, \ldots, M$, where the value of i is the *level* of the search. On each level, we maintain a collection of structures $(\mathcal{S}, T)$ with corresponding counters. On Level 0, there is only one structure $(\emptyset, V_0)$ with counter value 1.

The structures on Level i are extended in all possible ways, using edges from $\{\{v_1, v_2\} : v_1 \in V_i, v_2 \in V_{i+1}\}$ to get the structures for Level $i+1$. For each unordered pair in $\mathcal{S}$ and each vertex of the pair, we add one edge incident to the vertex. For each vertex in T, we add two distinct edges incident to the vertex. Details about this extension will be given in the next subsection. For each structure on Level $i+1$ obtained in this way, the corresponding counter is updated by adding the value of the counter of the structure from which the search started.

Since each vertex is incident to two edges of a Hamiltonian cycle, one can calculate in advance the number of edges of the Hamiltonian cycle between two consecutive sets of vertices. Namely, if one denotes the number of such edges between V_i and V_{i+1} by N_i ($N_{-1} = 0$ and $N_M = 0$), then we have that

$$N_{i-1} + N_i = 2|V_i|, \quad 0 \le i \le M,$$

and N_i can easily be obtained for all i.

2.2. **The extension step.** Extending the structures on Level i to structures on Level $i+1$ can be done in a backtrack search, adding edges with endpoints in V_i and V_{i+1}. We shall formulate the extension problem when going from Level i to Level $i+1$ within the following setting, which is a variant of the *exact cover problem*: Given a finite set U, a multiset $\mathcal{C}$ of subsets of U, and a function $f : U \to \mathbb{Z}^+$, find all submultisets $\mathcal{C}' \subseteq \mathcal{C}$ with the property that any element $u \in U$ occurs in $f(u)$ sets in $\mathcal{C}'$. The libexact software [15] can be used to solve instances of this problem.

Given an endpoint structure $(\mathcal{S}, T)$ on Level i, we now let $U = S' \cup T$ (recall that S' contains the elements in the sets in $\mathcal{S}$), $f(u) = 1$ for $u \in S'$, and $f(u) = 2$ for $u \in T$. Moreover, for every edge $\{v_1, v_2\} \in \mathcal{E}$ with $v_1 \in U$ and $v_2 \in V_{i+1}$, we put $\{v_1\}$ in $\mathcal{C}$ (and we must maintain information about the related edge).

During the search, one must keep track of the endpoints of paths that are formed and, in particular, make sure that no cycles are created. Note, however, that Level M is a special case; then exactly one cycle should be formed. All this can be achieved by maintaining a data structure that updates the endpoints of a path. Whenever an edge is added so that a vertex $v \in V_{i+1}$ becomes an internal vertex of a path, any sets in $\mathcal{C}$ corresponding to edges incident to v should be removed. Moreover, whenever a path is extended to get endpoints $v_1 \in U \subseteq V_i$ and $v_2 \in V_{i+1}$, a set in $\mathcal{C}$ corresponding to the edge $v_1 v_2$ can be removed.

Example 1. Consider the graph in Figure 1. This graph is a member of a family of graphs to be considered in Section 3. We partition the vertices as $V_0 = \{a\}$, $V_1 = \{b, c, d\}$, $V_2 = \{e, f, g\}$, and $V_3 = \{h\}$.

On Level 0, we start from $(\mathcal{S}, T) = (\emptyset, V_0)$. There are three ways to extend this path of length 0, leading to the following structures on Level 1: $(\{\{b, c\}\}, \{d\})$, $(\{\{b, d\}\}, \{c\})$, $(\{\{c, d\}\}, \{b\})$, all with occurrence count 1.

On Level 1, consider the structure $(\{\{b, c\}\}, \{d\})$. The vertex d must be connected to two vertices, f and g. Now for the four ways of connecting the vertices b and c, the choices be, ce and bf, cg lead to cycles, and the other two choices give

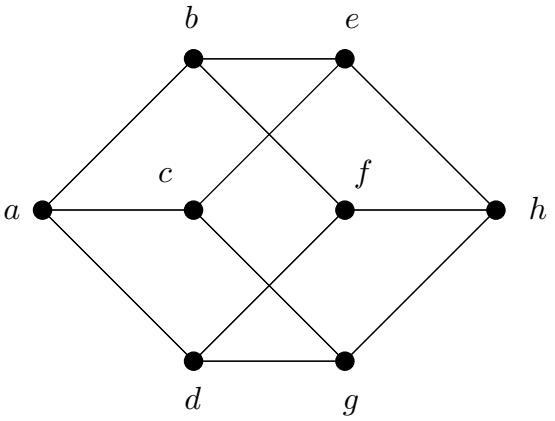

FIGURE 1. Example graph

the structures $(\{\{e, f\}\}, \emptyset)$ and $(\{\{e, g\}\}, \emptyset)$ on Level 2. By considering also the other two structures on Level 1, we get the structures $(\{\{e, f\}\}, \emptyset)$, $(\{\{e, g\}\}, \emptyset)$, and $(\{\{f, g\}\}, \emptyset)$ on Level 2, all with occurrence count 2.

All three structures on Level 2 can be completed to Level 3 in a unique way, so there are $2 + 2 + 2 = 6$ Hamiltonian cycles in the example graph.

The general description of the approach is now complete. We shall next look at various possibilities of speeding up the computations.

2.3. **A bidirectional approach, or gluing.** The outlined algorithm builds up parts of Hamiltonian cycles by starting from V_0. Obviously, one could as well start from the other direction, first considering the vertices in V_M (the necessary modifications to the approach are obvious). In particular, one can consider *both* of these simultaneously in a *bidirectional* approach (the term "meet-in-the-middle" also describes the setting).

If there is an automorphism in $\mathrm{Aut}(\Gamma)$ that maps the elements of V_i to V_{M-i} for all i, then the basic algorithm also gives structures for the opposite direction and is implicitly a bidirectional algorithm. Note that there may be several possible automorphisms; the chosen automorphism must be used consistently.

We call the process of combining structures obtained in a bidirectional approach *gluing*. A similar idea has been used earlier, for example, in [25]. Consider gluing of a structure $(\mathcal{S}_1, T_1)$ for $\bigcup_{j=0}^{M'} V_j$ with a structure $(\mathcal{S}_2, T_2)$ for $\bigcup_{j=M'}^{M} V_j$. We must then have $T_1 = V_{M'} \setminus (S_2' \cup T_2)$ and $T_2 = V_{M'} \setminus (S_1' \cup T_1)$; also note that $S_1' = S_2'$. For $\mathcal{S}_2$, on the other hand, there may be several choices given $\mathcal{S}_1$.

There are two obvious approaches for finding counterparts when gluing. Either one determines proper structures of counterparts and checks whether they exist, or one checks existing counterparts for proper structure. In the first case, given a structure $(\mathcal{S}_1, T_1)$, one may view $\mathcal{S}_1$ as a perfect matching over S_1' and obtain $\mathcal{S}_2$ as the edge sets that extend this perfect matching to a Hamiltonian cycle over S_1'. We denote the number of pairs in $\mathcal{S}_1$ by s. Now the Hamiltonian cycle over S_1' can be constructed by starting from one edge in $\mathcal{S}_1$ and iteratively extending the cycle by adding an edge to one of its endpoints. There are then $2(s-1)$ ways to get the

first pair in $\mathcal{S}_2$, $2(s-2)$ to get the second, and so on, making a total of $(s-1)!2^{s-1}$ proper structures of counterparts.

Example 2. Assume that gluing takes part in the set $V_{M'} = \{k, l, m, n, o, p\}$. Then the structure $(\{\{k, l\}, \{n, p\}\}, \{m\})$ can be glued with $(\{\{k, n\}, \{l, p\}\}, \{o\})$ and $(\{\{k, p\}, \{l, n\}\}, \{o\})$. Indeed $s = 2$, so $(s-1)!2^{s-1} = 2$.

In the second approach, it clearly suffices to check counterparts $(\mathcal{S}_2, T_2)$ with $S_2' = S_1'$ and $T_2 = V_{M'} \setminus (S_1' \cup T_1)$.

To optimize time usage, experimentation is required to evaluate the various possibilities (there are obviously many alternative implementations of the data structures). To obtain the number of Hamiltonian cycles, the values of the two counters for the parts involved in gluing are multiplied and the results are summed up.

The existence of an automorphism $\alpha \in \mathrm{Aut}(\Gamma)$ that maps the elements of V_i to V_{M-i} gives an opportunity for speeding up the overall algorithm by a factor of approximately 2: whenever two parts can be glued, α maps the two parts into two further parts that can be glued (we omit technical details regarding implementation of this idea).

Gluing can also be carried out before the parts of the bidirectional search have met. For instance, in [25] this technique of gluing is applied. In this case, further edges need to be added in the process of gluing.

2.4. **Counting with symmetry.** A nontrivial automorphism group $\mathrm{Aut}(\Gamma)$ is often helpful for speeding up graph algorithms. In Section 2.3 we have already seen one situation where an automorphism of order 2 is beneficial. A further situation where the current approach benefits from a nontrivial automorphism group will be discussed here.

We distinguish two types of counting results. In the first case, we count the *total number* of Hamiltonian cycles. In the second case, we count the *equivalence classes* of Hamiltonian cycles under the group $\mathrm{Aut}(\Gamma)$.

We denote the subgroup of a group G that stabilizes the set (or, more generally, set system) C by $\mathrm{Aut}_G(C)$. If the graph Γ is obvious from the context, we use the abbreviated form $\mathrm{Aut}(C) := \mathrm{Aut}_{\mathrm{Aut}(\Gamma)}(C)$ for the group of symmetries of a Hamiltonian cycle C in Γ, and call this the *automorphism group* of the Hamiltonian cycle.

With a nontrivial stabilizer subgroup

$$G' = \mathrm{Aut}_{\mathrm{Aut}(\Gamma)}((V_0, V_1, \ldots, V_M)), \tag{1}$$

the extension step of the basic algorithm can be modified as follows: for each structure $(\mathcal{S}, T)$, a canonical form under the action of G' is determined and the counter of that canonical form is updated. There is then a single counter for all structures with the same canonical form. The reader is referred to [14, Chapter 3] for further information on computing canonical forms of combinatorial structures.

Example 3. In the graph in Figure 1, G' as defined in (1) is isomorphic to the symmetric group S_3; any permutation of the vertices in V_1 and V_2 are in this group. Thereby, when the approach is applied to this graph, the three structures on Level 1 are in the same orbit under the action of G' and have the same canonical form, say $(\{b, c\}, \{d\})$, which in the new setting is the unique structure on Level 1 with occurrence count 3. The two possible extensions to Level 2 also have the same canonical form, say $(\{e, f\}, \emptyset)$, which gets occurrence count $3 + 3 = 6$. The unique

completion of this structure to Level 3 gives a total of six Hamiltonian cycles, as in Example 1.

A final value of N for the counter of a canonical form implies that any structure with the canonical form $(\mathcal{S}_1, T_1)$ has an individual counter value of

$$\frac{N|\mathrm{Aut}_{G'}((\mathcal{S}_1, T_1))|}{|G'|}. \tag{2}$$

In the extension step we may ignore this formula, since extending the canonical form with counter value N gives the same counter values on the next level as when extending

$$\frac{|G'|}{|\mathrm{Aut}_{G'}((\mathcal{S}_1, T_1))|} \tag{3}$$

structures whose counter values are given by (2).

When gluing, the counter for the total number of Hamiltonian cycles is updated in the following way. For each (canonical structure) $(\mathcal{S}_1, T_1)$, we find all possible structures $(\mathcal{S}_2, T_2)$ with which we can glue (as described in Section 2.3). Consider one such pair of structures and let the corresponding counters have values N and N', respectively.

The counter values for the two structures are (2) and

$$\frac{N'|\mathrm{Aut}_{G'}((\mathcal{S}_2, T_2))|}{|G'|},$$

respectively. Moreover, the number of structures with the canonical form $(\mathcal{S}_1, T_1)$ is given by (3). The total number of Hamiltonian cycles whose path structure in $\bigcup_{j=0}^{M'} V_j$ has the canonical form $(\mathcal{S}_1, T_1)$ is then

$$\frac{|G'|}{|\mathrm{Aut}_{G'}((\mathcal{S}_1, T_1))|} \cdot \frac{N|\mathrm{Aut}_{G'}((\mathcal{S}_1, T_1))|}{|G'|} \cdot \frac{N'|\mathrm{Aut}_{G'}((\mathcal{S}_2, T_2))|}{|G'|} = \frac{NN'|\mathrm{Aut}_{G'}((\mathcal{S}_2, T_2))|}{|G'|}.$$

It should be emphasized that the given formula is not symmetric with respect to the parts of gluing as $|\mathrm{Aut}_{G'}((\mathcal{S}_1, T_1))|$ may differ from $|\mathrm{Aut}_{G'}((\mathcal{S}_2, T_2))|$.

The choice of partition $(V_0, V_1, \ldots, V_M)$ obviously has an impact also here, via the stabilizer subgroup. For large random graphs, no partitions with a nontrivial stabilizer subgroup are to be expected, but some of the instances studied in the literature lead to such symmetry—to a small or a large extent. We shall now proceed by looking at a type of graph whose automorphism group is quite large, namely the n-cube.

3. Hamiltonian cycles in the n-cube

The *n-dimensional hypercube*, or *n-cube*, can be formed by taking a vertex for each binary vector of length n and adding edges between vertices whenever the corresponding vectors differ in exactly one coordinate. The n-cube is bipartite.

The problem of counting the Hamiltonian cycles in n-cubes with small n has attracted much attention, and has even been popularized [8]. Gilbert [10] studied the cases with $n \leq 4$: for $n = 2$, 3, and 4, there are, respectively, 1, 6, and 1 344 Hamiltonian cycles, and 1, 1, and 9 equivalence classes of Hamiltonian cycles. For

TABLE 1. Structures on Level 3, up to equivalence

Paths	#
1	13 495
2	263 305
3	2 782 510
4	17 003 576
5	61 154 671
6	127 360 225
7	142 398 993
8	65 084 556
9	7 887 199
10	139 098
Total	424 087 628

the 5-cube, the number of Hamiltonian cycles, 906 545 760, was obtained by Steve Winker [34] and confirmed by Bell and Hallowell [1]. The number of equivalence classes for the 5-cube, 237 675, was apparently first given by Knuth in the draft [16, p. 85] of [17, p. 688] without details; in-depth studies of this case were later published by Culver and Leach [3], and Dejter and Delgado [5] (an erroneous value was earlier mentioned in the survey [11]). The algorithms developed in the current work were tested against these old results.

The problem of counting the Hamiltonian cycles in the 6-cube is mentioned in [17, Exercise 7.2.1.1.43]; we shall next compute this number. We shall also classify the cycles with nontrivial automorphisms, from which the total number of equivalence classes will be obtained.

3.1. **The 6-cube.** We consider the case of the n-cube in general and the 6-cube in particular. We partition the vertices of the n-cube so that V_i consists of the vertices whose corresponding binary vectors have i 1's (in other words, they have Hamming weight i). The stabilizer subgroup (1), which is central in our approach, then has order $n!$, with elements corresponding to the permutations of the n coordinates. Apart from implementation details, we now have all we need for applying the approach described in Section 2.

For the 6-cube, there is one structure on Level 1 (counter value 15), up to equivalence:

$$(\{\{100000, 010000\}\}, \{001000, 000100, 000010, 000001\}).$$

On Level 2, there are 3 446 structures, up to equivalence. On Level 3, there is a combinatorial explosion in the number of structures; the numbers are summarized in Table 1 based on their numbers of paths.

Addition of the all-one vector gives an automorphism of the n-cube that maps V_i to V_{n-i}. Moreover, this enables gluing on Level $n/2$, that is, Level 3 for the 6-cube. The results of gluing on Level 3 are summarized in Table 2. The calculations took just under 10 core-years for our fastest implementation, using a PC cluster with 20 i7-870 processors and a total of 80 cores; roughly two-thirds of this time was spent gluing the structures with nine paths on Level 3. The results were validated using two independent implementations.

TABLE 2. Glued Hamiltonian cycles in the 6-cube

Paths	#
1	269 635 088 041 094 880
2	19 221 791 375 622 767 040
3	361 924 641 407 769 994 080
4	2 623 087 675 470 868 439 040
5	8 443 693 910 745 312 544 800
6	12 696 602 985 718 261 583 040
7	8 812 957 118 756 042 697 120
8	2 606 036 710 760 600 434 560
9	268 829 026 417 644 883 200
10	5 590 226 830 719 432 960
Total	35 838 213 722 570 883 870 720

The core of the algorithm is the determination of canonical forms. Tuning this part of the algorithm is essential for optimizing the speed of the calculations. However, since the part involves a lot of problem-specific bit-twiddling with little general interest, we have decided to omit the details here. Moreover, this part can be done in many ways.

By Table 2, we now have the following result.

Theorem 1. *The total number of undirected Hamiltonian cycles in the* 6*-cube is* 35 838 213 722 570 883 870 720*.*

Silverman, Vickers, and Sampson [29] used a ten-run series of experiments to estimate the number in Theorem 1. Their estimate was $4.6 \cdot 10^{24}/2^7 \approx 3.6 \cdot 10^{22}$, which is highly accurate.

3.2. **Hamiltonian cycles with nontrivial automorphisms.** The topic of classifying combinatorial structures with prescribed automorphisms has been extensively studied; see [13] and [14, Chapter 9] for more information and references to earlier studies. For a recent study of counting structures with prescribed automorphisms, see [12]. We shall now look at the problem of classifying Hamiltonian cycles with prescribed automorphisms in the n-cube, with the 6-cube as a running example.

It is convenient that the symmetries of a Hamiltonian cycle can be considered both as a subgroup of the automorphism group of the graph $\Gamma = (V, \mathcal{E})$ (here, the n-cube) and as a subgroup of the automorphism group of the $|V|$-cycle, the dihedral group $D_{|V|}$ (here, $|V| = 2^n$).

The automorphism group of the n-cube is isomorphic to the wreath product $S_2 \wr S_n$, which has order $2^n n!$. In the sequel, we refer to this group by G. If the vertices of the n-cube are identified with binary vectors of length n, as described earlier, then an automorphism can be presented as a permutation of the coordinates followed by the addition of a vector. We shall now determine which of these automorphisms can be automorphisms of a Hamiltonian cycle in the n-cube. By Cauchy's theorem, it suffices to focus on automorphisms of prime order.

Lemma 1. *A Hamiltonian cycle in the* n*-cube cannot have an automorphism of prime order greater than* 2*.*

Proof. By the earlier discussion, the automorphism group of the Hamiltonian cycle is a subgroup of $D_{|V|}$. Since the latter group has order 2^{n+1}, the order of a subgroup divides 2^{n+1} and cannot contain any prime factors but 2. □

Corollary 1. *The automorphism group of a Hamiltonian cycle in the n-cube with nontrivial automorphisms is a 2-group, that is, of order 2^j for some $j \geq 1$.*

Let us now have a look at what automorphisms of order 2 are possible. To reduce the number of cases to be studied, we notice that it suffices to study one automorphism from each conjugacy class in G. We number the coordinates of binary vectors of length n, corresponding to the vertices of the n-cube, by $1, 2, \ldots, n$. Up to conjugacy, an automorphism of order 2 consists of a transpositions of coordinates $(1\ 2)(3\ 4)\cdots$, followed by b coordinates whose value is complemented and c coordinates that are unchanged. Clearly, $n = 2a + b + c$. The letters a, b, and c will be used in this meaning throughout the rest of this work.

Lemma 2. *If $n > 2$, then no automorphism of order 2 fixes a vertex of a Hamiltonian cycle in the n-cube.*

Proof. Assume the existence of an automorphism of order 2 that fixes vertices of the n-cube. Since such an automorphism must not complement any values, $b = 0$. With given values of a and c, the number of fixed vertices is then 2^{a+c}. As any nontrivial automorphism can fix at most two vertices of a Hamiltonian cycle, it follows that $2^{a+c} \leq 2$, that is, $a + c \leq 1$. Hence $n = 2a + c \leq 2$. □

Corollary 2. *If $n > 2$, then any automorphism of order 2 of a Hamiltonian cycle in the n-cube has $b > 0$.*

Lemma 3. *No automorphism of order 2 of a Hamiltonian cycle in the n-cube has odd $b \geq 3$.*

Proof. Assume that $n \geq 3$. A Hamiltonian cycle in the n-cube is a cycle of length 2^n. An automorphism of order 2 of such a cycle either rotates the cycle 2^{n-1} positions or reflects the cycle. If the cycle is reflected, then either a vertex or an edge is fixed by the reflection. By Lemma 2, no vertex can be fixed, so two edges are fixed. An automorphism that fixes an edge maps a vertex to another vertex at distance 1, which cannot happen if $b \geq 2$.

Since the n-cube is bipartite and 2^{n-1} is even, an automorphism that rotates the cycle 2^{n-1} positions maps a vertex to a vertex that is at even distance in the n-cube. Then we cannot have b odd, since such an automorphism maps a vertex to a vertex at odd distance in the n-cube. □

We present one detail of the proof of Lemma 3 separately.

Lemma 4. *An automorphism of order 2 cannot fix more than two edges of a Hamiltonian cycle in the n-cube.*

Lemma 5. *The order of an automorphism of a Hamiltonian cycle in the n-cube is at most $2^{1+\lfloor \log_2 n \rfloor}$.*

Proof. Any automorphism of the n-cube can be represented as a permutation over a set of $2n$ elements. If the permutation is written in cyclic form, with cycles of length $l_1, l_2, \ldots$, then the order of the automorphism is $\operatorname{lcm}(l_1, l_2, \ldots)$, the value of which is 2^j for some j by Corollary 1. Consequently, all l_j are powers of 2, and the order of the automorphism is $\max(l_1, l_2, \ldots) \leq 2n$. □

Corollary 3. *The order of the automorphism group of a Hamiltonian cycle in the n-cube is at most $2^{2+\lfloor \log_2 n \rfloor}$.*

Proof. This follows from Lemma 5 as the automorphism group of a Hamiltonian cycle is a cyclic group or a dihedral group. □

We shall now use Corollary 2 and Lemma 3 to reduce the list of automorphisms of order 2 to be considered in the case of a 6-cube. We denote complementing a coordinate m by $(\overline{m})$. Up to conjugacy, it suffices to consider the following cases:

Case 1: $(1\ 2)(3\ 4)(\overline{5})(\overline{6})$
Case 2: $(1\ 2)(3\ 4)(\overline{5})$
Case 3: $(1\ 2)(\overline{3})(\overline{4})(\overline{5})(\overline{6})$
Case 4: $(1\ 2)(\overline{3})(\overline{4})$
Case 5: $(1\ 2)(\overline{3})$
Case 6: $(\overline{1})(\overline{2})(\overline{3})(\overline{4})(\overline{5})(\overline{6})$
Case 7: $(\overline{1})(\overline{2})(\overline{3})(\overline{4})$
Case 8: $(\overline{1})(\overline{2})$
Case 9: $(\overline{1})$

For each automorphism listed here, one can now search for all Hamiltonian cycles that admit such automorphisms. However, although this search can be done considerably faster than a search for all Hamiltonian cycles—as we search for sets of orbits rather than sets of edges—an exhaustive search is still challenging for $n = 6$. Therefore we approach this problem via perfect matchings.

A *perfect matching* in a graph is a set of edges with the property that every vertex of the graph is incident to exactly one edge in the set. It is obvious that by taking every second edge of a Hamiltonian cycle, we get a perfect matching. A recent theoretical result (which we do not need here) in fact says that the converse holds for n-cubes: every perfect matching in the n-cube can be extended to a Hamiltonian cycle [7]. The following lemma is essential for implementing this approach with prescribed automorphisms.

Lemma 6. *Let $n \geq 3$. The automorphisms of a Hamiltonian cycle in the n-cube stabilizes the two perfect matchings formed by taking every second edge of the cycle.*

Proof. An automorphism of a Hamiltonian cycle in the n-cube is either a reflection or a rotation. For a rotation to map one perfect matching onto the other, it must involve an odd number of steps, s. However, since then $\gcd(s, 2^n) = 1$, the order of such an automorphism would be 2^n, which by Lemma 5 is not possible when $n \geq 3$.

As argued in the proof of Lemma 3, any reflection for $n \geq 3$ fixes two edges. Then all other edges in the same perfect matching M are necessarily mapped to other edges in M. □

There are fast algorithms for finding all perfect matchings of a bipartite graph [30]. However, we here solve such instances in the (suboptimal) framework of exact cover as (a) this is not the most time-consuming part of the overall algorithm, (b) the libexact software [15] can be used directly for such instances, and (c) few modifications are necessary when considering prescribed automorphisms.

Using the notation from Section 2.2, the problem of finding all perfect matchings in a graph $\Gamma = (V, \mathcal{E})$ can be formulated as finding all solutions of the exact cover problem with $U = V$, $\mathcal{C} = \mathcal{E}$, and $f(u) = 1$ for all $u \in U$.

With a prescribed automorphism of order 2, we now solve an instance of the exact cover problem, where the candidate subsets are orbits (of one or two edges) rather than edges. Additional pruning can (and should) be carried out during the search: by Lemma 4 it suffices to consider solutions that have at most two orbits of size 1. Such pruning is essential when there is a large number of candidate orbits of size 1, as in Case 9.

The solutions obtained after prescribing each of the nine automorphisms are processed as follows. We denote the group of order 2 generated by the prescribed automorphism by H. The perfect matchings obtained are subject to isomorph rejection relative to the normalizer

$$N_G(H) := \{g \in G : gHg^{-1} = H\}.$$

This part is implemented via an encoding to a graph and using the graph isomorphism program *nauty* [24].

In Table 3, we give the number of representatives obtained for each of the nine cases, partitioned with respect to the order of the $N_G(H)$-automorphism group, $\mathrm{Aut}_{N_G(H)}(M)$. It should be emphasized that, due to the additional pruning, the tabulated values do *not* represent a complete classification of perfect matchings with prescribed automorphisms.

As a consistency check, for each case and group order it was verified that that the total number of solutions obtained equals the number of representatives multiplied by

$$\frac{|N_G(H)|}{|\mathrm{Aut}_{N_G(H)}(M)|} = \frac{a!b!c!2^n}{|\mathrm{Aut}_{N_G(H)}(M)|}. \tag{4}$$

For each of the perfect matchings classified, we continue the search with the same prescribed automorphism for the rest of the edges. Obviously, the sets of edges we are searching for form perfect matchings, so the exact cover approach can be used in this stage as well. Moreover, the exact cover search can now be pruned if a cycle arises before the search is completed.

For isomorph rejection of the Hamiltonian cycles obtained, *nauty* can obviously be used, but there is also the possibility of using a certain basic representation of the cycles that makes it possible to develop an alternative faster algorithm. A *delta sequence* [17, p. 293] is a sequence of 2^n numbers between 1 and n that expresses the coordinates that are altered, traversing the cycle in one of the directions from some vertex. With a given starting point and direction, it is easy to transform the delta sequence into a canonical form by requiring that the first occurrence of i come before that of j iff $i < j$. For a given Hamiltonian cycle, it is then straightforward to compute a canonical form and the order of the automorphism group, and to determine whether there is a reflection in the automorphism group (no attention is paid to the original prescribed automorphism here).

We use the established terminology and call the perfect matchings that the final search starts from *seeds* [14, p. 188]. The collection of seeds (with their prescribed automorphisms) is denoted by $\mathcal{P}$.

TABLE 3. Certain perfect matchings in the 6-cube

|Aut|\Case	1	2	3	4	5
2	64 780	124 893	8 164	74 807	77 394
4	13 870	12 934	3 061	35 260	12 491
6			15		8
8	1 571	1 349	607	4 522	1 288
12			46		15
16	278	219	170	852	184
24			31		16
32	52	43	51	170	45
48			11		2
64	6	4	16	40	15
96			1		
128	4	4	3	8	2
192			3		2
384			1		
Total	80 561	139 446	12 180	115 659	91 462

|Aut|\Case	6	7	8	9
2	186	3 784	6 687	5 896
4	189	2 637	11 868	1 487
6		2	6	8
8	84	870	2 771	161
10	2			2
12	6	40	52	20
16	27	271	592	28
20	6			2
24	7	35	84	12
32	13	86	124	13
48	5	16	54	7
64	3	31	46	5
80	1			
96	5	8	22	3
128	4	8	6	2
192	2	3	4	1
256	1	4	4	
384	1	1	3	1
768		1	1	
1536	1	2	2	1
7680	1			
Total	544	7 799	22 326	7 649

The number of Hamiltonian cycles in the 6-cube with an automorphism group of order at least 4 is small enough that it is possible to tabulate (canonical forms of) such cycles for straightforward isomorph rejection.

For cycles with an automorphism group of order 2, however, an alternative approach is required. Consider such a Hamiltonian cycle C obtained starting from a perfect matching M. By the Orbit-Stabilizer theorem,

$$|\mathrm{Aut}_{N_G(H)}(M)|/|\mathrm{Aut}_{N_G(H)}(C)| = |\mathrm{Aut}_{N_G(H)}(M)|/2. \tag{5}$$

Hamiltonian cycles $\mathrm{Aut}_{N_G(H)}$-equivalent to C will be encountered in the search starting from M. Now, for each Hamiltonian cycle with an automorphism group of order 2 that we find, we add the inverse of (5) to a counter. We must further divide the final value of the counter by 2 as each Hamiltonian cycle can be obtained from either of the two perfect matchings that it contains.

The number of equivalence classes for different orders of the automorphism group are tabulated in Table 4. The total computation time for all stages of this classification was just over one core-week using a contemporary PC.

TABLE 4. Hamiltonian cycles with nontrivial automorphisms in the 6-cube, up to equivalence

$\lvert\mathrm{Aut}\rvert$\Type	All	Reflected
2	7 001 923 981	4 369 328 232
4	220 165	195 606
8	568	494
16	20	20
Total	7 002 144 734	4 369 524 352

As in the first stage, we use a double counting argument for validation. For all Hamiltonian cycles C and all automorphisms $\pi \in \mathrm{Aut}(C)$ of order 2, we now want to count the pairs (C, π) in two different ways. Such an approach is closely related to the topic of [12] and is essentially an application of the Orbit-Stabilizer theorem.

Our collection of representatives from the equivalence classes of Hamiltonian cycles with nontrivial automorphisms is denoted by $\mathcal{C}$. The desired count can be obtained by

$$\sum_{C \in \mathcal{C}} \frac{|G| T_C}{|\mathrm{Aut}(C)|}, \tag{6}$$

where T_C denotes the number of automorphisms of order 2 in $\mathrm{Aut}(C)$. If $|\mathrm{Aut}(C)| = 2$ or the automorphism group is a cyclic group, then $T_C = 1$; otherwise, the automorphism group is a dihedral group D_m, $m \geq 2$—in other words, there are symmetries that reflect the cycle—and $T_C = m + 1$. (Recall that we indeed get all necessary information in the delta sequence computations.)

An alternative way of obtaining the count is to sum

$$\frac{1}{2} \sum_{P \in \mathcal{P}} \frac{|N_G(H_P)|}{|\mathrm{Aut}_{N_G(H_P)}(P)|} \cdot \frac{|G|}{|N_G(H_P)|} \cdot V_P = \frac{1}{2} \sum_{P \in \mathcal{P}} \frac{|G| V_P}{|\mathrm{Aut}_{N_G(H_P)}(P)|}, \tag{7}$$

where V_P denotes the total number of solutions found in the search starting from the seed P and H_P is the prescribed group related to P. The reason for dividing by 2 in (7) is that the two involved perfect matchings will be formed in both orders, so each object would otherwise be counted twice. The classification can now be validated by checking that (6) = (7).

Note that the described technique is here effective explicitly for Hamiltonian cycles with an automorphism group of order at least 4, as the algorithm relies on a counting argument (5) rather than a classification for cycles with an automorphism group of order 2. For the case of automorphism groups of order 2, the following alternative validation method—based on a technique called *canonical augmentation* (see [14, Section 4.2.3] and [26])—can be utilized.

The canonical form of a Hamiltonian cycle—obtained in the delta sequence computations—can be used to put the two involved perfect matchings into a canonical order. Indeed, by Lemma 6 it is always possible to distinguish between the two perfect matchings. In the search, we maintain two counters for Hamiltonian cycles with an automorphism group of order 2, and, for every such cycle found, choose the counter to update based on whether the seed is the canonically smaller perfect matching or not. At the end of the search, the two counts should be equal.

The number of Hamiltonian cycles with no nontrivial automorphisms can now be obtained by the Orbit-Stabilizer theorem,

$$N = \sum_i \frac{|G| N_i}{i} = \sum_i \frac{2^n n! N_i}{i}, \tag{8}$$

where N is the total number of Hamiltonian cycles and N_i is the number of equivalence classes of Hamiltonian cycles with an automorphism group of order i. Applying (8) to the values in Theorem 1 (which gives N) and Table 4, we get that $N_1 = 777\,739\,009\,575\,607\,980$ for the 6-cube. It then follows that the total number of equivalence classes is

$$\sum_i N_i = 777\,739\,016\,577\,752\,714.$$

All instances of n-cubes with $n < 6$ were also processed in a similar manner.

A recent manuscript [6] on arXiv claims, without details, that the number of Hamiltonian cycles in the 6-cube is 14 754 666 508 334 433 250 560 and that the corresponding number of equivalence classes is 147 365 405 634 413 085. It is easy to see that at least one of the numbers is in error, since their ratio should be smaller than $|G| = 2^6 6! = 46\,080$—in fact, it should be very close to this value, as only a small fraction of the cycles have nontrivial automorphisms—but it is approximately 100 000.

One may further count the number of Hamiltonian cycles with a given direction of the cycle. For the total number, this is exactly two times the number of undirected cycles. However, when considering equivalence classes, each class gives either one or two equivalence classes of directed cycles, depending on whether the automorphism group contains reflections or not. If there are reflections, so that only one equivalence class is obtained, then the order of the automorphism group of the directed Hamiltonian cycle is half the order for the original undirected cycle; otherwise the order of the group is unchanged. The number of equivalence classes of directed Hamiltonian cycles in the 6-cube is tabulated in Table 5.

TABLE 5. Directed Hamiltonian cycles in the 6-cube, up to equivalence

|Aut|	#
1	1 555 478 023 520 544 192
2	5 265 387 104
4	49 612
8	168
Total	1 555 478 028 785 981 076

4. Knight's tours on a chessboard

The problem of counting the number of knight's tours on an 8×8 chessboard can be viewed as the problem of counting the Hamiltonian cycles in a graph with one vertex for each square of the board and edges corresponding to legal moves of a knight. This graph is obviously bipartite; the color of the square (black or white) always changes when a knight moves. A method for counting the desired number is presented in [22], but the number claimed in [22] turned out to be incorrect and was later corrected by McKay in [25].

This graph demonstrates that the best partition into sets V_i is not always clear. Inspired by the results for the n-cube, one might let V_0 consist of the vertex corresponding to one of the corner squares, and let V_i consist of the vertices at distance i from that vertex. Unfortunately, this does not lead to desired symmetries. Two partitions of the chessboard squares that do lead to a nice setting for gluing are shown in Figure 2. The areas to be glued can in both cases be mapped onto each other by a 180-degree rotational symmetry.

0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2
2	3	2	3	2	3	2	3
3	2	3	2	3	2	3	2
4	3	4	3	4	3	4	3
3	4	3	4	3	4	3	4

0	1	2	1	2	3	2	3
1	2	1	2	3	2	3	4
2	1	2	3	2	3	4	3
1	2	3	2	3	4	3	4
2	3	2	3	4	3	4	5
3	2	3	4	3	4	5	4
2	3	4	3	4	5	4	5
3	4	3	4	5	4	5	6

Figure 2. Partitions of vertices

Based on experiments, the first partition was used for calculations in this work. A good general heuristic for choosing partitions seems to be that of minimizing $\max |V_i|$.

For the first partition in Figure 2, the number of structures is 1, 143 379, and 95 345 608 on levels 0, 1, and 2, respectively. Gluing gives 13 267 364 410 532 knight's tours, which corroborates the result in [25].

5. Conclusions

Application of the described techniques to nonbipartite graphs has been briefly mentioned in this paper; further research is required to tune the details and evaluate the performance and usefulness of such algorithms. A generalization to directed graph is also possible—this is indeed a generalization since an undirected graph can be viewed as a particular directed graph with every edge replaced by two arcs, one in each direction.

Acknowledgments

The authors are grateful to Donald Knuth and the anonymous referees for valuable comments, and to the Department of Special Collections at Stanford University Libraries for providing a copy of [34]. This work was financed by the Academy of Finland under Grants No. 110196, 130142, and 132122. The research of the second

author was partly carried out while he was visiting Universität Bayreuth; Dr. Axel Kohnert and Professors Reinhard Laue and Michael Stoll are gratefully acknowledged for their hospitality during this visit.

References

[1] A. Bell and P. Hallowell, *Crawling round a cube edge*, Computing (U.K.) (Feb. 1973), 9.
[2] Nicos Christofides, *Graph theory: An algorithmic approach*, Computer Science and Applied Mathematics, Academic Press [Harcourt Brace Jovanovich Publishers], New York, 1975. MR0429612 (55 #2623)
[3] Clay Culver and David Leach, *Equivalence classes of 5-bit Gray codes*, Australas. J. Combin. **39** (2007), 129–134. MR2351193 (2008g:05008)
[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to algorithms*, 3rd ed., MIT Press, Cambridge, MA, 2009. MR2572804 (2010j:68001)
[5] Italo J. Dejter and Abel A. Delgado, *Classes of Hamilton cycles in the 5-cube*, J. Combin. Math. Combin. Comput. **61** (2007), 81–95. MR2322204 (2008i:05109)
[6] M. Deza and R. Shklyar, Enumeration of Hamiltonian cycles in 6-cube, arXiv:1003.4391v1, 2010.
[7] Jiří Fink, *Perfect matchings extend to Hamilton cycles in hypercubes*, J. Combin. Theory Ser. B **97** (2007), no. 6, 1074–1076, DOI 10.1016/j.jctb.2007.02.007. MR2354719 (2008g:05120)
[8] M. Gardner, Mathematical Games, Scientific American **227**(2) (Aug. 1972), 106–109.
[9] Michael R. Garey and David S. Johnson, *Computers and intractability*, A guide to the theory of NP-completeness; A Series of Books in the Mathematical Sciences, W. H. Freeman and Co., San Francisco, Calif., 1979. MR519066 (80g:68056)
[10] E. N. Gilbert, *Gray codes and paths on the n-cube*, Bell System Tech. J **37** (1958), 815–826. MR0094273 (20 #792)
[11] Frank Harary, John P. Hayes, and Horng-Jyh Wu, *A survey of the theory of hypercube graphs*, Comput. Math. Appl. **15** (1988), no. 4, 277–289, DOI 10.1016/0898-1221(88)90213-1. MR949280 (89i:05230)
[12] Alexander Hulpke, Petteri Kaski, and Patric R. J. Östergård, *The number of Latin squares of order 11*, Math. Comp. **80** (2011), no. 274, 1197–1219, DOI 10.1090/S0025-5718-2010-02420-2. MR2772119 (2011m:05064)
[13] Petteri Kaski, *Isomorph-free exhaustive generation of designs with prescribed groups of automorphisms*, SIAM J. Discrete Math. **19** (2005), no. 3, 664–690 (electronic), DOI 10.1137/S0895480104444788. MR2191287 (2006h:05038)
[14] Petteri Kaski and Patric R. J. Östergård, *Classification algorithms for codes and designs*, Algorithms and Computation in Mathematics, vol. 15, Springer-Verlag, Berlin, 2006. With 1 DVD-ROM (Windows, Macintosh and UNIX). MR2192256 (2008a:05002)
[15] P. Kaski and O. Pottonen, libexact user's guide, version 1.0, HIIT Technical Reports 2008-1, Helsinki Institute for Information Technology HIIT, 2008.
[16] Donald E. Knuth, *The art of computer programming. Vol. 4, Fasc. 2*, Generating all tuples and permutations, Addison-Wesley, Upper Saddle River, NJ, 2005. MR2251595 (2007f:68004b)
[17] D. E. Knuth, *The Art of Computer Programming, Volume 4A, Combinatorial Algorithms, Part 1*, Addison-Wesley, Upper Saddle River, 2011. MR0378456 (51:14624)
[18] Donald E. Knuth, *Selected papers on fun & games*, CSLI Lecture Notes, vol. 192, CSLI Publications, Stanford, CA, 2011. MR2905906 (2012m:01013)
[19] William Kocay, *An extension of the multi-path algorithm for finding Hamilton cycles*, Discrete Math. **101** (1992), no. 1-3, 171–188, DOI 10.1016/0012-365X(92)90601-B. Special volume to mark the centennial of Julius Petersen's "Die Theorie der regulären Graphs", Part II. MR1172376 (93g:05095)
[20] William Kocay and Donald L. Kreher, *Graphs, algorithms, and optimization*, Discrete Mathematics and its Applications (Boca Raton), Chapman & Hall/CRC, Boca Raton, FL, 2005. MR2106632 (2005k:05001)
[21] D. L. Kreher and D. R. Stinson, *Combinatorial algorithms: Generation, enumeration, and search*, CRC Press, Boca Raton, 1999.

[22] Martin Loebbing and Ingo Wegener, *Comments on: "The number of knight's tours equals 33,439,123,484,294—counting with binary decision diagrams"*, Electron. J. Combin. **3** (1996), no. 1, Research Paper 5, Comment 1, 1 HTML document (electronic). MR1715429

[23] S. Martello, Algorithm 595: An enumerative algorithm for finding Hamiltonian circuits in a directed graph, *ACM Trans. Math. Software* **9** (1983), 131–138.

[24] B. D. McKay, *nauty* user's guide (version 1.5), Technical Report TR-CS-90-02, Computer Science Department, Australian National University, Canberra, 1990.

[25] B. D. McKay, Knight's tours of an 8×8 chessboard, Technical Report TR-CS-97-03, Computer Science Department, Australian National University, Canberra, 1997.

[26] Brendan D. McKay, *Isomorph-free exhaustive generation*, J. Algorithms **26** (1998), no. 2, 306–324, DOI 10.1006/jagm.1997.0898. MR1606516 (98k:68132)

[27] Frank Rubin, *A search procedure for Hamilton paths and circuits*, J. Assoc. Comput. Mach. **21** (1974), 576–580. MR0349480 (50 #1973)

[28] Jefferey A. Shufelt and Hans J. Berliner, *Generating Hamiltonian circuits without backtracking from errors*, Theoret. Comput. Sci. **132** (1994), no. 1-2, 347–375, DOI 10.1016/0304-3975(94)90239-9. MR1290549 (95e:05067)

[29] Jerry Silverman, Virgil E. Vickers, and John L. Sampson, *Statistical estimates of the n-bit Gray codes by restricted random generation of permutations of 1 to* 2^n, IEEE Trans. Inform. Theory **29** (1983), no. 6, 894–901, DOI 10.1109/TIT.1983.1056755. MR733197 (85c:94030)

[30] Takeaki Uno, *A fast algorithm for enumerating bipartite perfect matchings*, Algorithms and computation (Christchurch, 2001), Lecture Notes in Comput. Sci., vol. 2223, Springer, Berlin, 2001, pp. 367–379, DOI 10.1007/3-540-45678-3_32. MR1917757 (2003e:05130)

[31] Leslie G. Valiant, *The complexity of enumeration and reliability problems*, SIAM J. Comput. **8** (1979), no. 3, 410–421, DOI 10.1137/0208032. MR539258 (80f:68055)

[32] B. Vandegriend, *Finding Hamiltonian Cycles: Algorithms, Graphs and Performance*, M.Sc. thesis, Department of Computing Science, University of Alberta, Edmonton, 1998.

[33] Douglas B. West, *Introduction to graph theory*, Prentice Hall Inc., Upper Saddle River, NJ, 1996. MR1367739 (96i:05001)

[34] S. Winker to M. Gardner, 31 October 1972, *Guide to the Martin Gardner Papers*, Department of Special Collections, Stanford University Libraries, Stanford.

Department of Communications and Networking, Aalto University School of Electrical Engineering, P.O. Box 13000, 00076 Aalto, Finland

Department of Communications and Networking, Aalto University School of Electrical Engineering, P.O. Box 13000, 00076 Aalto, Finland