Trimming and Gluing Gray Codes[*][†]

Petr Gregor[1] and Torsten Mütze[2]

1 Department of Theoretical Computer Science and Mathematical Logic, Charles University, Praha, Czech Republic
gregor@ktiml.mff.cuni.cz

2 Institut für Mathematik, TU Berlin, Berlin, Germany
muetze@math.tu-berlin.de

Abstract

We consider the algorithmic problem of generating each subset of $[n] := \{1, 2, \ldots, n\}$ whose size is in some interval $[k, l]$, $0 \leq k \leq l \leq n$, exactly once (cyclically) by repeatedly adding or removing a single element, or by exchanging a single element. For $k = 0$ and $l = n$ this is the classical problem of generating all 2^n subsets of $[n]$ by element additions/removals, and for $k = l$ this is the classical problem of generating all $\binom{n}{k}$ subsets of $[n]$ by element exchanges. We prove the existence of such cyclic minimum-change enumerations for a large range of values n, k, and l, improving upon and generalizing several previous results. For all these existential results we provide optimal algorithms to compute the corresponding Gray codes in constant time $\mathcal{O}(1)$ per generated set and space $\mathcal{O}(n)$. Rephrased in terms of graph theory, our results establish the existence of (almost) Hamilton cycles in the subgraph of the n-dimensional cube Q_n induced by all levels $[k, l]$. We reduce all remaining open cases to a generalized version of the middle levels conjecture, which asserts that the subgraph of Q_{2k+1} induced by all levels $[k - c, k + 1 + c]$, $c \in \{0, 1, \ldots, k\}$, has a Hamilton cycle. We also prove an approximate version of this conjecture, showing that this graph has a cycle that visits a $(1 - o(1))$-fraction of all vertices.

1998 ACM Subject Classification G.2.1 Combinatorics, G.2.2 Graph Theory

Keywords and phrases Gray code, subset, combination, loopless algorithm

Digital Object Identifier 10.4230/LIPIcs.STACS.2017.40

1 Introduction

Generating all objects in a combinatorial class such as permutations, subsets, combinations, partitions, trees, strings etc. is one of the oldest and most fundamental algorithmic problems, and such generation algorithms appear as core building blocks in a wide range of practical applications (see the survey [30]). In fact, half of the most recent volume [21] of Donald Knuth's seminal series *The Art of Computer Programming* is devoted entirely to this fundamental subject. The ultimate goal for algorithms that efficiently generate each object of a particular combinatorial class exactly once is to generate each new object in constant time, which is best possible. Such optimal algorithms are sometimes called *loopless algorithms*, a term coined by Ehrlich in his influential paper [10]. Note that a constant-time algorithm requires in particular that consecutively generated objects differ only in a constant amount, e.g., in a single transposition of a permutation, in adding or removing a single element from a set, or in a single tree rotation operation. These types of orderings have become known as

[*] A full version of the paper is available at https://arxiv.org/abs/1607.08806.
[†] The first author was supported by the Czech Science Foundation grant GA14-10799S.

combinatorial Gray codes. Here are two fundamental examples for this kind of generation problems: (1) The so-called *reflected Gray code* is a method to generate all 2^n many subsets of $[n] := \{1, 2, \ldots, n\}$ by repeatedly adding or removing a single element. It is named after Frank Gray, a physicist and researcher at Bell Labs, and appears in his patent [13]. The reflected Gray code has many interesting properties (see [21, Section 7.2.1.1]), and there is a simple loopless algorithm to compute it [2, 10]. (2) Of similar importance in practice is the problem of generating all $\binom{n}{k}$ many k-element subsets of $[n]$ by repeatedly exchanging a single element. Also for this problem, loopless algorithms are well-known [2, 4, 8, 9, 10, 18, 28, 35] (see also [21, Section 7.2.1.3]).

In this work we consider far-ranging generalizations of the classical problems (1) and (2). Specifically, we consider the algorithmic problem of generating all (or almost all) subsets of $[n]$ whose size is in some interval $[k, l]$, $0 \leq k \leq l \leq n$, by repeatedly adding or removing a single element, or by exchanging a single element (this will made more precisely in a moment). We recover the classical problems (1) and (2) mentioned before as special cases by setting $k = 0$ and $l = n$, or by setting $k = l$, respectively. It turns out that the entire parameter range in between these special cases offers plenty of room for surprising discoveries and hard research problems (take a peek at the known results in Figure 1 below).

In a computer a subset of $[n]$ is conveniently represented by the corresponding characteristic bitstring x of length n, where all the 1-bits of x correspond to the elements contained in the set, and the 0-bits to the elements not contained in the set. The before-mentioned subset generation problems can thus be rephrased as Hamilton cycle problems in subgraphs of the *cube* Q_n, the graph that has as vertices all bitstrings of length n, with an edge between any two vertices (=bitstrings) that differ in exactly one bit. We refer to the number of 1-bits in a bitstring x as the *weight of* x, and we refer to the vertices of Q_n with weight k as the *k-th level* of Q_n (there are $\binom{n}{k}$ vertices on level k). Moreover, we let $Q_{n,[k,l]}$, $0 \leq k \leq l \leq n$, denote the subgraph of Q_n induced by all levels $[k, l]$. In terms of sets, the vertices of the cube Q_n correspond to subsets of $[n]$, and flipping a bit along an edge corresponds to adding or removing a single element. The weight of a bitstring corresponds to the size of the set, and the vertices on level k correspond to all k-element subsets of $[n]$.

One of the hard instances of the before-mentioned general enumeration problem in $Q_{n,[k,l]}$ is when $n = 2k+1$ and $l = k+1$. The existence of a Hamilton cycle in the graph $Q_{2k+1,[k,k+1]}$ for any $k \geq 1$ is asserted by the well-known *middle levels conjecture*, raised independently in the 80's by Havel [16] and Buck and Wiedemann [3]. The conjecture has also been attributed to Dejter, Erdős, Trotter [20] and various others, and also appears in the popular books [5, 21, 36]. The middle levels conjecture has attracted considerable attention over the last 30 years [6, 7, 12, 15, 17, 19, 20, 27, 29, 31, 32, 33], and a positive solution, an existence proof for a Hamilton cycle in $Q_{2k+1,[k,k+1]}$ for any $k \geq 1$, has been announced only recently.

▶ **Theorem 1** ([23]). *For any $k \geq 1$, the graph $Q_{2k+1,[k,k+1]}$ has a Hamilton cycle.*

The following generalization of the middle levels conjecture was proposed in [15].

▶ **Conjecture 2** ([15]). *For any $k \geq 1$ and $c \in \{0, 1, \ldots, k\}$, the graph $Q_{2k+1,[k-c,k+1+c]}$ has a Hamilton cycle.*

Conjecture 2 clearly holds for all $k \geq 1$ and $c = k$ as $Q_{2k+1,[0,2k+1]} = Q_{2k+1}$ (this is problem (1) from before). It is known that the conjecture also holds for all $k \geq 1$ and $c = k - 1$ [11, 22] and $c = k - 2$ [15]. By Theorem 1 it also holds for all $k \geq 1$ and $c = 0$.

Another generalization of Theorem 1 in a slightly different direction (still a special case in our general framework) is the following result.

▶ **Theorem 3** ([26]). *For any $n \geq 3$ and $k \in \{1, 2, \ldots, n-2\}$, the graph $Q_{n,[k,k+1]}$ has a cycle that visits all vertices in the smaller bipartite class.*

The idea for the proof of Theorem 3 (induction over n) was first presented in [16]. In that paper, the theorem was essentially proved conditional on the validity of the hardest case $n = 2k + 1$, the middle levels conjecture, which was established as a theorem (Theorem 1) only much later. In [26], Theorem 3 was proved unconditionally, and the proof technique was refined further to also prove Hamiltonicity results for the so-called bipartite Kneser graphs, another generalization of the middle levels conjecture.

Conjecture 2 and Theorem 3 immediately suggest the following common generalization: For which intervals $[k, l]$ does the cube $Q_{n,[k,l]}$ have a Hamilton cycle? The graph $Q_{n,[k,l]}$ is bipartite (the two partition classes are given by the parity of the number of 1-bits of the vertices), and it is clear that a Hamilton cycle can exist only if the two partition classes have the same size, which happens only for odd dimension n and between two symmetric levels k and $l = n - k$ (Conjecture 2). However, we may slightly relax this question, and ask for a long cycle. To this end, we denote for any bipartite graph G by $v(G)$ the number of vertices of G, and by $\delta(G)$ the difference between the larger and the smaller partition class. Note that in any bipartite graph G (as $Q_{n,[k,l]}$) the length of any cycle is at most $v(G) - \delta(G)$ (i.e., the cycle visits all vertices in the smaller partition class). We call such a cycle a *saturating cycle*. Observe that if both partition classes have the same size (i.e. $\delta(G) = 0$), then a saturating cycle is a Hamilton cycle. Hence saturating cycles naturally generalize Hamilton cycles for unbalanced bipartite graphs. The right common generalization of Conjecture 2 and Theorem 3 therefore is:

▶ Question 4. For which intervals $[k, l]$ does the cube $Q_{n,[k,l]}$ have a saturating cycle?

A saturating cycle necessarily omits some vertices (exactly $\delta(Q_{n,[k,l]})$ many) from the larger bipartite class. However, if we insist on all vertices of $Q_{n,[k,l]}$ to be included in a cycle, then this can be achieved by allowing steps where instead of only a single bitflip, two bits are flipped (the underlying graph $Q_{n,[k,l]}$ is augmented by adding distance-2 edges). In this case we may ask for a (cyclic) enumeration of all vertices of $Q_{n,[k,l]}$ that minimizes the number of these 'cheating' distance-2 steps, i.e., for an enumeration that has only $\delta(Q_{n,[k,l]})$ many distance-2 steps (this is clearly the least number one can hope for). We call such an enumeration a *tight enumeration*. A tight enumeration can be seen as a travelling salesman tour of length $v(Q_{n,[k,l]}) + \delta(Q_{n,[k,l]})$ through all vertices of $Q_{n,[k,l]}$, where distances are measured by Hamming distance (=number of bitflips). We may ask in full generality:

▶ Question 5. For which intervals $[k, l]$ is there a tight enumeration of the vertices of $Q_{n,[k,l]}$?

If both partition classes of $Q_{n,[k,l]}$ have the same size (i.e. $\delta(Q_{n,[k,l]}) = 0$), then a tight enumeration is a Hamilton cycle in this graph. Note also that Question 5 is a sweeping generalization of the following well-known result regarding problem (2) mentioned before.

▶ **Theorem 6** ([35]). *For any $n \geq 2$ and $1 \leq k \leq n-1$ there is a cyclic enumeration of all weight k bitstrings of length n such that any two consecutive bitstrings differ in exactly 2 bits.*

In fact, several of the subsequently mentioned results of this paper will be proved by extending the original approach from [35] to prove Theorem 6.

1.1 Our results

In this work we answer Question 4 and Question 5 for a large range of values n, k and l. The different ranges of parameters covered by our results are illustrated in Figure 1.

Moreover, we provide optimal algorithms to compute the corresponding saturating cycles/tight enumerations.

Our first set of results resolves Question 4 positively for all possible values of k and l except the cases covered by Conjecture 2 (the case $l = k+1$ is already covered by Theorem 3), see the left-hand side of Figure 1.

▶ **Theorem 7.** *For any $n \geq 3$ the graph $Q_{n,[k,l]}$ has a saturating cycle in the following cases:*

(i) *If $0 = k < l \leq n$ or $0 \leq k < l = n$ and $l - k \geq 2$.*
(ii) *If $1 \leq k < l \leq n-1$ and $l - k \geq 2$ is even.*
(iii) *If $1 \leq k < l \leq \lceil n/2 \rceil$ or $\lfloor n/2 \rfloor \leq k < l \leq n-1$ and $l - k \geq 3$ is odd.*
(iv) *If $1 \leq k < l \leq n-1$ and $l - k \geq 3$ is odd, under the additional assumption that $Q_{2m+1,[m-c,m+1+c]}$, $c := (l-k-1)/2$, has a Hamilton cycle for all $m = c, c+1, \ldots, (\min(k+l, 2n-k-l)-1)/2$.*

Our second set of results resolves Question 5 positively for all possible values of k and l except the cases covered by Conjecture 2 (the case $l = k$ is already covered by Theorem 6), see the right-hand side of Figure 1.

▶ **Theorem 8.** *For any $n \geq 3$ there is a tight enumeration of the vertices of $Q_{n,[k,l]}$ in the following cases:*

(i) *If $0 = k < l \leq n$ or $0 \leq k < l = n$.*
(ii) *If $1 \leq k < l \leq n$ and $l - k \geq 2$ is even.*
(iii) *If $1 \leq k < l \leq n-1$ and $l - k = 1$.*
(iv) *If $1 \leq k < l \leq \lceil n/2 \rceil$ or $\lfloor n/2 \rfloor \leq k < l \leq n-1$ and $l - k \geq 3$ is odd.*
(v) *If $1 \leq k < l \leq n-1$ and $l - k \geq 3$ is odd, under the additional assumption that $Q_{2m+1,[m-c,m+1+c]}$, $c := (l-k-1)/2$, has a Hamilton cycle for all $m = c, c+1, \ldots, (\min(k+l, 2n-k-l)-1)/2$.*

Note that the last part (iv) of Theorems 7 and 8 is conditional on the validity of Conjecture 2. In fact, by what we said before (recall the paragraph below Conjecture 2) we know that the additional assumption in (iv) is satisfied for $m \in \{c, c+1, c+2\}$ (so the statement could be slightly strengthened).

The tight enumerations we construct to prove Theorem 8 have the additional property that all distance-2 steps are within single levels (but never between two different levels k and $k+2$). In terms of sets, these steps therefore correspond to exchanging a single element.

For all the unconditional results in Theorems 7 and 8 we provide corresponding optimal generation algorithms.

▶ **Theorem 9.**

(a) *For any interval $[k,l]$ as in case (i) or (ii) of Theorems 7 and 8, respectively, there is a corresponding loopless algorithm that generates each bitstring of a saturating cycle or a tight enumeration of the vertices of $Q_{n,[k,l]}$ in time $\mathcal{O}(1)$.*
(b) *For any interval $[k,l]$ as in case (iii) of Theorems 7 and 8, respectively, there is a corresponding algorithm that generates each bitstring of a saturating cycle or a tight enumeration of the vertices of of $Q_{n,[k,l]}$ in time $\mathcal{O}(1)$ on average.*

It should be noted that the algorithms for part (a) of Theorem 9 are considerably simpler than those for part (b). The reason is that the underlying constructions are entirely different. In particular, for part (b) we repeatedly call the (average) constant-time algorithm to compute a Hamilton cycle in $Q_{2m+1,[m,m+1]}$, $m \leq \lfloor (n-1)/2 \rfloor$, an algorithmic version of Theorem 1,

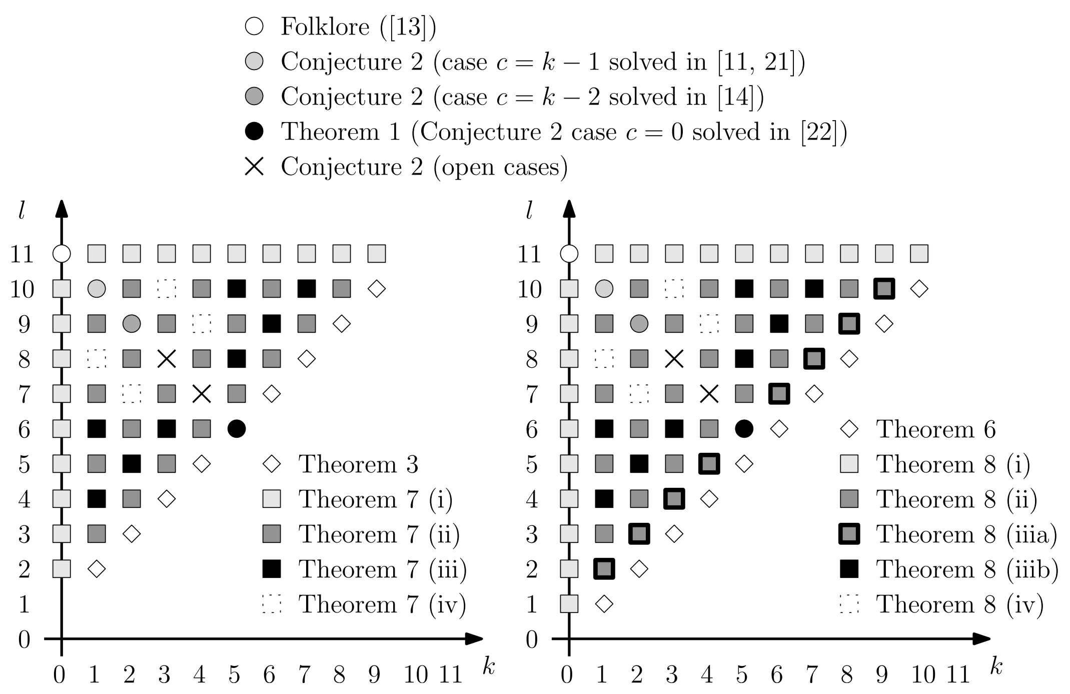

Figure 1 The different cases of k and l covered by our two main theorems on saturating cycles (left, Theorem 7) and on tight enumerations (right, Theorem 8) in $Q_{n,[k,l]}$ for the case $n = 11$. A more extensive animation of the entire parameter space (n, k, l) is available on the second author's website [1].

presented in [24, 25] (and this algorithm is admittedly rather complex). The initialization time of our algorithms is $\mathcal{O}(n)$, and the required space is $\mathcal{O}(n)$.

We implemented all these algorithms in C++, and we invite the reader to experiment with this code, which can be found on our website [1].

In view of these results, the only remaining (and therefore even more interesting) open case is the question whether the cube of odd dimension has a Hamilton cycle between any two symmetric levels, i.e., Conjecture 2 (these open cases are represented by crosses in Figure 1). Given the results from [15] and [23], the next natural step towards resolving this conjecture would be to investigate whether the graphs $Q_{2k+1,[3,2k-2]}$ or $Q_{2k+1,[k-1,k+2]}$ have a Hamilton cycle for all $k \geq 1$.

In this work we provide the following partial result towards the general conjecture: We show the existence of long cycles in the graph $Q_{2k+1,[k-c,k+1+c]}$, $c \in \{0, 1, \ldots, k\}$. This approximate version of the conjecture is similar in spirit to the line of work [12, 19, 29, 31] that preceded the proof of Theorem 1.

▶ **Theorem 10.** *For any $k \geq 1$ and $c \in \{0, 1, \ldots, k\}$, the graph $Q_{2k+1,[k-c,k+1+c]}$ has a cycle that visits at least a $(1-\epsilon)$-fraction of all vertices, where $\epsilon := \frac{1}{2(c+1)} \min\big(1, \exp(\frac{(c+1)^2}{k-c}) - 1\big)$. In particular, for any c and $k \to \infty$, the cycle visits a $(1 - o(1))$-fraction of all vertices.*

1.2 Related work

In [10] an algorithm is presented that generates the vertices of $Q_{n,[k,l]}$ (for an arbitrary interval $[k, l]$) such that any two consecutive vertices have Hamming distance 1 or 2, where the value 2 appears only between vertices on level k and l, but the Hamming distance between the first and last vertex is arbitrary (possibly n). The running time of this algorithm is $\mathcal{O}(n)$ per generated vertex. In addition, this paper presents a loopless algorithm (time $\mathcal{O}(1)$ per vertex) to generate all vertices in $Q_{n,[k,l]}$ level by level, using only distance-2 steps in each level. In particular, these enumerations are not cycles in $Q_{n,[k,l]}$, and they are not tight.

In [34] the authors present algorithms for enumerating all vertices of $Q_{n,[k,l]}$ (for an arbitrary interval $[k,l]$) such that any two consecutive bitstrings have Levenshtein distance at most 2 and Hamming distance at most 4. The Levenshtein distance is the minimum number of bit insertions, deletions, and bitflips necessary to transform one bitstring into the other. Again, these enumerations are not cycles in $Q_{n,[k,l]}$ and they are not tight. However, the corresponding generation algorithms are very simple and fast (loopless). Improving on this, as a byproduct of the results mentioned in the previous section we obtain a simple loopless algorithm to enumerate all vertices of $Q_{n,[k,l]}$ (for an arbitrary interval $[k,l]$) such that any two consecutive bitstrings have Hamming distance (and Levenshtein distance) at most 2.

1.3 Outline of this paper

In this extended abstract we restrict ourselves to only explaining the main ideas behind our constructions, algorithms and proofs. Lengthy pseudocode and proofs of auxiliary statements are omitted. Specifically, in Sections 2 and 3 we present our constructions for proving Theorems 7 and 8, respectively. In Section 4 we present our algorithm for part (a) of Theorem 9. The algorithm for part (b) and the proof of Theorem 10 as well as other omitted proofs can be found in the full version [14].

2 Saturating cycles

2.1 Trimming Gray codes and proof of Theorem 7 (i)+(ii)

In this section we sketch how to prove cases (i) and (ii) from Theorem 7 by showing that the standard reflected Gray code in Q_n mentioned in the introduction (see [13] and [21, Section 7.2.1.1]) can be 'trimmed' to any number of consecutive levels of Q_n so that it visits all these vertices except possibly some vertices from the first and last levels. This technique is a generalization of the approach presented in [35] to prove Theorem 6, and it yields the following result:

▶ **Theorem 11.** *For any $n \geq 3$ and k, l with $0 \leq k < l \leq n$ and $l - k \geq 2$, the graph $Q_{n,[k,l]}$ has a cycle that visits all vertices except possibly some vertices from levels k and l.*

Note that if $l - k$ is even, then the first and last level of $Q_{n,[k,l]}$ are from the same bipartite class, so the cycle obtained from Theorem 11 is saturating, which immediately yields Theorem 7 (ii).

The *(n-dimensional) reflected Gray code* Γ_n is a (cyclic) sequence Γ_n of all vertices of Q_n defined recursively by

$$\Gamma_1 = (0,1) \ , \tag{1a}$$

$$\Gamma_{n+1} = (\Gamma_n \circ 0, \mathrm{rev}(\Gamma_n) \circ 1) \ , \quad n \geq 1 \ , \tag{1b}$$

where $\Gamma_n \circ 0$ and $\Gamma_n \circ 1$ denote the sequences obtained from Γ_n by attaching a 0-bit or 1-bit to every entry of Γ_n, respectively, and $\mathrm{rev}(\Gamma_n)$ denotes the reversed sequence. See Figure 2 for an illustration.

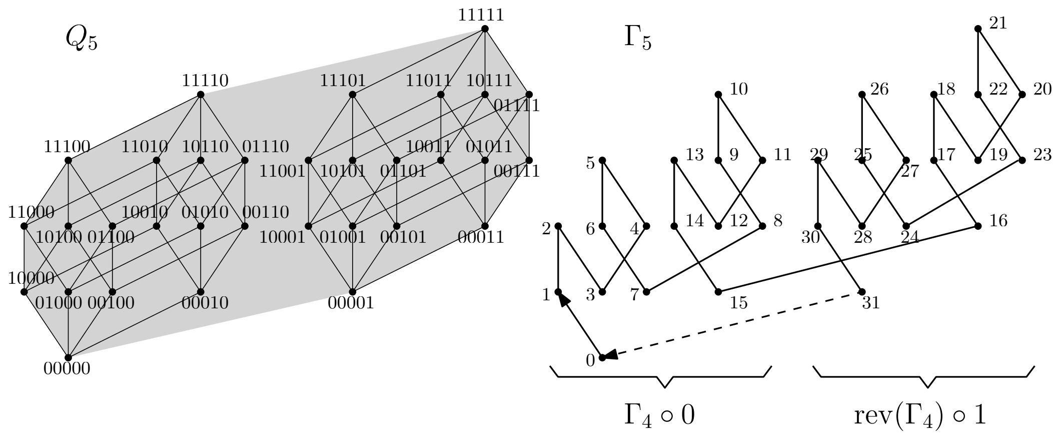

Figure 2 The hypercube Q_5 (left), where the grey area represents all 16 edges along which the last bit is flipped, and the reflected Gray code Γ_5 in Q_5 (right), where the numbers are indices of vertices in Γ_5 (starting from 0). The dashed edge represents the adjacency between the last and first vertex of Γ_5.

The reflected Gray code Γ_n is the standard example how to enumerate all bitstrings of length n such that any two consecutive bitstrings differ in exactly one bit. For an explicit definition of Γ_n and further interesting properties see [21, Section 7.2.1.1].

For any $0 \leq k \leq n$ let $\Gamma_{n,k}$ be the subsequence of Γ_n that contains all vertices in level k. Moreover, we let $s_{\Gamma_{n,k}}(x) = s(x)$ denote the successor of x in $\Gamma_{n,k}$ (if x is the last vertex of $\Gamma_{n,k}$, then $s(x)$ is the first vertex of $\Gamma_{n,k}$). Similarly, let $s_{\Gamma_n}(x)$ denote the successor of x in Γ_n.

As already observed in [35], any two consecutive vertices in $\Gamma_{n,k}$ differ in exactly two positions. The sequence $\Gamma_{n,k}$ therefore provides an enumeration of all k-element subsets of $[n]$ such that any two consecutive k-sets differ in exchanging a single element (recall Theorem 6).

▶ **Lemma 12** ([35]). *For any $n \geq 2$ and $0 < k < n$ let x be a vertex of $\Gamma_{n,k}$ and $y := s_{\Gamma_{n,k}}(x)$. Then x and y differ in exactly 2 bits.*

Clearly, any two vertices x, y in level k at distance 2 have a unique common neighbor in level $k-1$ and a unique common neighbor in level $k+1$, let us denote them by $\text{down}(x, y)$ and $\text{up}(x, y)$, respectively. The key idea in trimming the reflected Gray code to a given sequence of consecutive levels $[k, l]$, where $l - k \geq 2$, is to replace the subpath P of Γ_n in Q_n between a vertex x in level $l-1$ and its consecutive vertex $s_{\Gamma_{n,l-1}}(x)$ by the path $(x, \text{up}(x, s(x)), s(x))$ if P ascends above level $l-1$, and between a vertex x in level $k+1$ and its consecutive vertex $s_{\Gamma_{n,k+1}}(x)$ by the path $(x, \text{down}(x, s(x)), s(x))$ if P descends below level $k+1$. See Figure 3 for an illustration.

Formally, we say that a vertex x of Q_n in level k is an *upward vertex* if $s_{\Gamma_n}(x)$ is in level $k+1$, and a *downward vertex* if $s_{\Gamma_n}(x)$ is in level $k-1$ (no other case is possible). Thus, the reflected Gray code Γ_n ascends from upward vertices and descends from downward vertices. For any $0 < k \leq n$, we let $\text{up}(\Gamma_{n,k})$ denote the sequence of all vertices $\text{up}(x, s(x))$ in level $k+1$, where x is an upward vertex of $\Gamma_{n,k}$, in the order induced by $\Gamma_{n,k}$. Similarly, for any $0 \leq k < n$ we let $\text{down}(\Gamma_{n,k})$ denote the sequence of all vertices $\text{down}(x, s(x))$ in level $k-1$, where x is a downward vertex of $\Gamma_{n,k}$, in the order induced by $\Gamma_{n,k}$.

The next lemma captures the key property guaranteeing that in trimming the reflected Gray code as described above we will never visit the same vertex twice. Note that

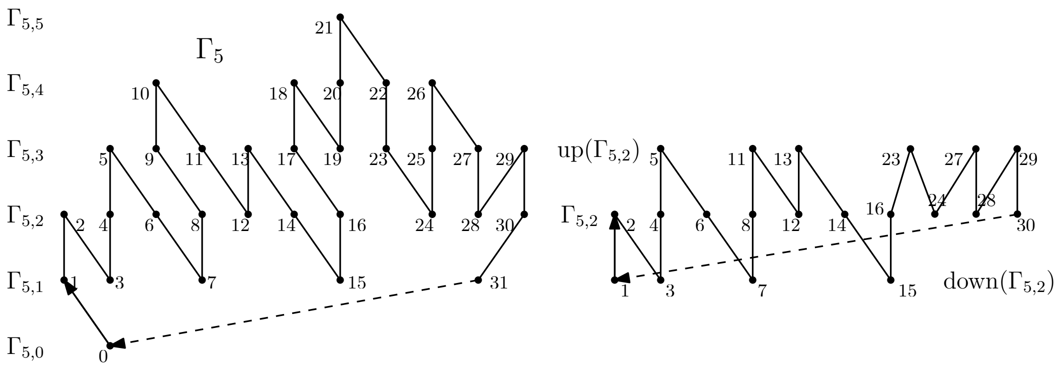

Figure 3 Schematic drawing of Γ_5 (left) highlighting the order in which levels are visited, and the corresponding sequences $\Gamma_{5,k}$, $0 \leq k \leq 5$. See Figure 2 what the actual vertices are. The sequences $\text{up}(\Gamma_{5,2})$, $\text{down}(\Gamma_{5,2})$, and Γ_5 trimmed to levels 1 up to 3 of Q_5 (right).

by a cyclic rotation of a sequence $(u_1, \ldots, u_{m-1}, u_m)$ to the right we mean the sequence $(u_m, u_1, \ldots, u_{m-1})$.

▶ **Lemma 13.** *For any $n \geq 2$ and $0 < k < n$, $\text{up}(\Gamma_{n,k})$ is a subsequence of $\Gamma_{n,k+1}$, and cyclically rotating $\text{down}(\Gamma_{n,k})$ once to the right yields a subsequence of $\Gamma_{n,k-1}$.*

With Lemma 13 in hand we are ready to prove Theorem 11.

Proof of Theorem 11. We build the desired cycle by trimming the reflected Gray code Γ_n to levels $[k,l]$. Subpaths of Γ_n within the levels $[k+1, l-1]$ remain unchanged (including the orientation). Each subpath P of Γ_n that descends from some downward vertex x at level $k+1$ to lower levels returns back to level $k+1$ at the vertex $y := s_{\Gamma_{n,k+1}}(x)$. As x and y differ in exactly 2 bits by Lemma 12, we may replace P by the path $P' = (x, \text{down}(x,y), y)$. Note that P' has the same end vertices and orientation as P, and visits only a single vertex at level k. After trimming all these descending paths, we visit on level k precisely the vertices of $\text{down}(\Gamma_{n,k+1})$. Since $\text{down}(\Gamma_{n,k+1})$ (after one rotation to the right) is a subsequence of $\Gamma_{n,k}$ by Lemma 13, all these vertices are distinct, and hence distinct trimmed paths may have at most end vertices in level k in common.

Similar arguments apply for trimming subpaths of Γ_n ascending from upward vertices at level $l-1$ to levels above. In this case the trimmed subpaths visit at level l precisely the vertices of $\text{up}(\Gamma_{n,l-1})$, and since this is a subsequence of $\Gamma_{n,l}$ by Lemma 13, all these vertices are distinct. Therefore trimming correctly produces a cycle visiting all vertices in levels $[k,l]$ except the vertices from level k that are not in $\text{down}(\Gamma_{n,k+1})$ and the vertices from level l that are not in $\text{up}(\Gamma_{n,l-1})$. ◀

Proof of Theorem 7 (ii). Follows immediately from Theorem 11, using that if $l-k$ is even, then the first and last level of $Q_{n,[k,l]}$ are from the same bipartite class. ◀

Proof of Theorem 7 (i). We only consider the case $0 = k < l \leq n$, the other case follows by symmetry, using that $Q_{n,[k,l]}$ is isomorphic to $Q_{n,[n-l,n-k]}$. The proof proceeds very similarly to the proof of Theorem 11, but we only trim the subpaths of Γ_n ascending above level $l-1$, so that the highest level where vertices are visited is level l (no trimming is applied at the bottom level 0). We therefore obtain a cycle that visits all vertices in levels $[0,l]$, except the vertices from level l that are not in $\text{up}(\Gamma_{n,l-1})$. As the cycle omits only vertices from level l, it must be saturating. ◀

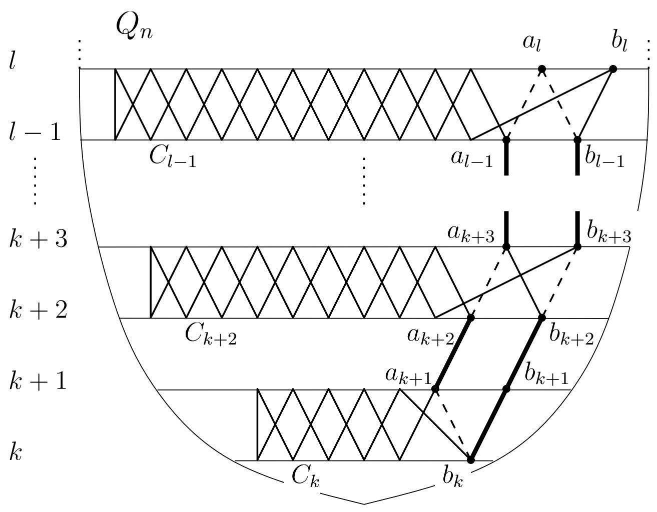

Figure 4 Notations used in the proof of Theorem 7 (iii). The removed edges are dashed, the added edges are bold.

2.2 Gluing saturating cycles and proof sketch of Theorem 7 (iii)+(iv)

Trimming the reflected Gray code Γ_n to levels $[k,l]$ as described in the last section does not yield a saturating cycle when $l-k \geq 3$ is odd unless $k=0$ or $l=n$. In general the trimmed cycle omits some vertices from both levels k and l, which are from different bipartite classes for odd $l-k$. We therefore use a different strategy to prove Theorem 7 (iii): We glue together several saturating cycles obtained from Theorem 3 (see Figure 4 for an illustration). To be able to join the cycles across different levels, each cycle between levels i and $i+1$ is first transformed by applying a suitable bit permutation (an automorphism of $Q_{n,[i,i+1]}$), in such a way that the permuted cycle visits certain distinguished vertices a_i and b_i in levels i and $i+1$ in a certain order, and so that it omits certain other vertices. The cycles are then glued together by removing one or two edges from each cycle and by joining the resulting paths by adding a few other cube edges. The details of this construction, in particular the definition of the vertices a_i and b_i, can be found in the full version [14]. This gluing approach yields a saturating cycle only if all involved levels are either below or above the middle (this is reflected by the conditions $l \leq \lceil n/2 \rceil$ or $\lfloor n/2 \rfloor \leq k$ in Theorem 7 (iii)). Otherwise the omitted vertices would be from different bipartite classes, so the resulting cycle would not be saturating. To prove Theorem 7 (iv), we inductively glue together pairs of saturating cycles of smaller dimension. Both proofs are similar to the approach presented in [26].

3 Tight enumerations

We call a sequence C that contains each vertex of $Q_{n,[k,l]}$ exactly once an *enumeration of the vertices of* $Q_{n,[k,l]}$ (recall that the successor of the last vertex of C equals the first vertex of C). The *total distance* of the enumeration C is $\mathrm{td}(C) := \sum_{u\in C} d(u, s_C(u))$, where $d(x,y)$ denotes the Hamming distance between x and y. As any two consecutive bitstrings in any enumeration have distance at least 1 and distance at least 2 if they are from the same bipartite class of $Q_{n,[k,l]}$, we have

$$\mathrm{td}(C) \geq v(Q_{n,[k,l]}) + \delta(Q_{n,[k,l]}) \tag{2}$$

(both the number of vertices $v(Q_{n,[k,l]})$ and the difference between the larger and smaller partition class $\delta(Q_{n,[k,l]})$ can be easily computed explicitly). A *tight enumeration* is an enumeration for which the lower bound (2) is attained. Clearly, an enumeration is tight, if and only if it has only distance-1 and distance-2 steps, and all the distance-2 steps are within the same partition class of $Q_{n,[k,l]}$ (this will be the larger partition class, and there will be exactly $\delta(Q_{n,[k,l]})$ such distance-2 steps in this class).

The proof of Theorem 8 is very similar to the proof of Theorem 7, and the key ideas in the different cases (i)–(iv) are the same. In this extended abstract, we only present the proofs for cases (i)–(ii). The proofs for cases (iii)–(iv) can be found in [14].

3.1 Proof of Theorem 8 (i)–(ii)

For $k = l$ we have $v(Q_{n,[k,k]}) = \delta(Q_{n,[k,k]}) = \binom{n}{k}$ and a tight enumeration of all weight k bitstrings of length n is given by $\Gamma_{n,k}$, by Lemma 12 (this is exactly the proof of Theorem 6 presented in [35]). We now proceed to prove cases (i)–(ii) of Theorem 8.

Proof of Theorem 8 (ii). We trim the reflected Gray code Γ_n to levels $[k,l]$, but in a slightly different fashion from the proof of Theorem 11. Specifically, subpaths of Γ_n within the levels $[k,l]$ remain unchanged (including the orientation). Moreover, each subpath P of Γ_n that descends from a downward vertex x in level k returns back to level k at the vertex $y := s_{\Gamma_{n,k}}(x)$, and we replace P by the distance-2 step (x, y) (recall Lemma 12). Similarly, each subpath P of Γ_n that ascends from an upward vertex x in level l returns back to level l at the vertex $y := s_{\Gamma_{n,l}}(x)$, and we replace P by the distance-2 step (x, y). This yields an enumeration C of all vertices of $Q_{n,[k,l]}$. Moreover, since $l - k$ is even, levels k and l of $Q_{n,[k,l]}$ belong to the same bipartite class, so all distance-2 steps of C are within the same bipartite class. It follows that C is a tight enumeration. ◀

Proof of Theorem 8 (i). We only consider the case $0 = k < l \leq n$, the other case follows by symmetry. The proof proceeds very similarly as the proof of part (ii), but we only trim the subpaths of Γ_n ascending above level l (no trimming is applied at the bottom level 0). This yields an enumeration C of all vertices of $Q_{n,[0,l]}$. Moreover, all distance-2 steps of C are within the same bipartite class (in level l), implying that C is a tight enumeration. ◀

4 A loopless algorithm for trimmed Gray codes

In this section we present a loopless algorithm to generate trimmed Gray codes, which is needed to prove part (a) of Theorem 9 (algorithms for cases (i) and (ii) of Theorems 7 and 8). The correctness proof and the runtime analysis for this algorithm can be found in the full version [14]. Also, the description of an efficient algorithm to compute glued Gray codes, which is needed to prove part (b) of Theorem 9 (algorithms for case (iii) of Theorems 7 and 8), can be found in [14].

Loopless algorithms both for the reflected Gray code Γ_n and for its restriction $\Gamma_{n,k}$ to one level of the cube (i.e, an algorithmic version of Theorem 6) were already provided in [2, 10]. However, these two algorithms cannot simply be merged into a loopless algorithm producing the trimmed Gray code. Instead, we provide a loopless algorithm that is based on an explicit description of successor vertices. This description is a simple (constant-time computable) rule describing which bit positions of a given vertex x from $\Gamma_{n,k}$ have to be flipped to reach the next vertex $s_{\Gamma_{n,k}}(x)$. In this extended abstract we do not explicitly state these rules, they are however used implicitly in the following pseudocode.

Algorithm 1: TrimGC(n, k, l)

Input: Integers $n \geq 2$ and $0 \leq k < l \leq n$, $l - k \geq 2$.

Result: The algorithm visits all vertices of the trimmed Gray code in $Q_{n,[k,l]}$ (which is a saturating cycle in cases (i) and (ii) of Theorem 7).

```
T1  $c := k+1$; $x := 1^c 0^{n-c}$;                    /* initialize current level $c$, vertex $x$ ... */
T2  $\nu_c := 1$; for $i := 1$ to $c$ do $p_i := c - i + 1$     /* ... and $(\nu_1, \ldots, \nu_c)$, $(p_1, \ldots, p_c)$ */
T3  while not enough vertices visited do
T4      if ($c$ is even $\wedge\ x_1 = 0$) $\vee$ ($c$ is odd $\wedge\ p_c < n \wedge x_{p_c+1} = 0$) then up := true else up := false
T5      if (up = true $\wedge\ c < l - 1$) then                          /* follow $\Gamma_n$ up */
T6          if $c$ is even then                                          /* $x_1 = 0$ */
T7              $x_1 := 1$; Visit$(x)$; $c := c + 1$; $p_c := 1$
T8              if $x_2 = 0$ then $\nu_c := c$ else $\nu_c := \nu_{c-1}$
T9          else                                             /* $c$ is odd and $x_{i+1} = 0$ */
T10             $i := p_c$; $x_{i+1} := 1$; Visit$(x)$; $c := c + 1$; $p_c := i$; $p_{c-1} := i + 1$
T11             if $(i + 1 = n\ \vee\ x_{i+2} = 0)$ then $\nu_c := c - 1$ else $\nu_c := \nu_{c-2}$
T12     else if (up = false $\wedge\ c > k + 1$) then                     /* follow $\Gamma_n$ down */
T13         if $c$ is even then                                          /* $x_1 = 1$ */
T14             $x_1 := 0$; Visit$(x)$; $c := c - 1$
T15             if $x_2 = 1$ then $\nu_c := \nu_{c+1}$
T16         else                                             /* $c$ is odd and $x_{i+1} = 1$ */
T17             $i := p_c$; $x_{i+1} := 0$; Visit$(x)$; $c := c - 1$; $p_c := i$; $\nu_c := c$
T18             if $x_{i+2} = 1$ then $\nu_{c-1} := \nu_{c+1}$
T19     else if (up=true $\wedge\ c = l - 1$) then        /* follow $\Gamma_{n,c}$ through up$(x, s_{\Gamma_{n,c}}(x))$ */
T20         if $c$ is even then                                /* $x_{i-1} = 0$ and $x_i = 1$ */
T21             $i := p_c$; $x_{i-1} := 1$; Visit$(x)$; $x_i := 0$; Visit$(x)$; $p_c := i - 1$
T22             if $x_{i+1} = 1$ then $\nu_{c-1} := \nu_c$; $\nu_c := c$
T23         else                                 /* $c$ is odd, $x_{i+1} = 0$ and $x_i = 1$ */
T24             $i := p_c$; $x_{i+1} := 1$; Visit$(x)$; $x_i := 0$; Visit$(x)$; $p_c := i + 1$
T25             if $(i + 2 \leq n\ \wedge\ x_{i+2} = 1)$ then $\nu_c := \nu_{c-1}$
T26     else            /* up = false $\wedge\ c = k + 1$; follow $\Gamma_{n,c}$ through down$(x, s_{\Gamma_{n,c}}(x))$ */
T27         if $x_1 = 0$ then $i := 1$ else $i := p_{\nu_c} + 1$          /* $i$ is minimal s.t. $x_i = 0$ */
T28         if $c \not\equiv i \bmod 2$ then                          /* $x_{i-2} = 1$ and $x_i = 0$ */
T29             $x_{i-2} := 0$; Visit$(x)$; $x_i := 1$; Visit$(x)$; $p_{\nu_c+1} := i - 1$; $p_{\nu_c} := i$
T30             if $(i + 1 \leq n\ \wedge\ x_{i+1} = 1)$ then $\nu_{\nu_c+1} := \nu_{\nu_c-1}$ else $\nu_{\nu_c+1} := \nu_c$
T31             if $i > 3$ then $\nu_c := \nu_c + 2$
T32         else                         /* $c \equiv i \bmod 2$; $j > i$ is minimal s.t. $x_j = 1$ */
T33             if $x_1 = 0$ then $a := c$; $j := p_a$ else $a := \nu_c - 1$; $j := p_a$
T34             if $j = n$ then $x_n := 0$; Visit$(x)$; $x_i := 1$; Visit$(x)$; $p_1 := i$; $\nu_c := 1$
T35             else if $x_{j+1} = 0$ then                   /* $j < n$, $x_{i-1} = 1$ and $x_{j+1} = 0$ */
T36                 $x_{i-1} := 0$; Visit$(x)$; $x_{j+1} := 1$; Visit$(x)$; $p_{\nu_c} := j$; $p_{\nu_c-1} := j + 1$
T37                 if $(j + 2 \leq n\ \wedge\ x_{j+2} = 1)$ then $\nu_{\nu_c} := \nu_{\nu_c-2}$ else $\nu_{\nu_c} := \nu_c - 1$
T38                 if $i > 2$ then $\nu_c := \nu_c + 1$
T39             else                                       /* $j < n$, $x_{j+1} = 1$ and $x_i = 0$ */
T40                 $x_{j+1} := 0$; Visit$(x)$; $x_i := 1$; Visit$(x)$; $p_a := i$; $p_{a-1} := j$
T41                 if $(j + 2 \leq n\ \wedge\ x_{j+2} = 1)$ then $\nu_{a-2} := \nu_a$
T42                 if $j = i + 1$ then $\nu_c := a - 1$ else $\nu_{a-1} := a - 1$, $\nu_c := a$
```

Consider now the algorithm TrimGC(n, k, l) that computes a trimmed Gray code between levels k and l with $l-k \geq 2$ of Q_n as described in Section 2.1 (Theorem 11), i.e., the algorithm produces a saturating cycle for cases (i) and (ii) of Theorem 7. At the end of this section we describe how to modify the algorithm to produce a tight enumeration for cases (i) and (ii) of Theorem 8.

The algorithm maintains the current vertex in the variable x and the current level in the variable c, $k < c < l$. Both are initialized in line T1 to $c = k+1$ and $x = 1^c 0^{n-c}$, where b^k denotes the k-fold repetition of the symbol $b \in \{0, 1\}$ (the code can easily be modified to start at a different vertex). The algorithm visits the sequence of vertices along the trimmed Gray code by the calls Visit(x), which could be some user-defined function that reads x. For simplicity the main while-loop does not have an explicit termination criterion, but this can be added easily (e.g., by providing an additional argument to the algorithm that specifies the number of vertices to be visited before termination). If the while-loop is not terminated, then the same cycle is traversed over and over again. There are four main cases distinguished in the while-loop of the algorithm: If the current level is between $k+1$ and $l-1$, then we either follow Γ_n by flipping a 0-bit to a 1-bit (if the condition in line T5 is satisfied), or we follow Γ_n by flipping a 1-bit to a 0-bit (if the condition in line T12 is satisfied). If the current level is $c = l-1$ and x is an upward vertex (the condition in line T19 is satisfied), then we first flip a 0-bit to the vertex up(x, y), $y := s_{\Gamma_{n,c}}(x)$, and then a 1-bit to the vertex y. On the other hand, if the current level is $c = k+1$ and x is a downward vertex (the condition in line T26 is satisfied), then we first flip a 1-bit to the vertex down(x, y), $y := s_{\Gamma_{n,c}}(x)$, and then a 0-bit to the vertex y.

The key to our loopless algorithm is to be able to determine in constant time the smallest integer $i \geq 1$ with $x_i = 1$ or with $x_i = 0$ (this part of the before-mentioned successor rule computation). Furthermore, we also need the smallest integer $j > i$ with $x_j = 1$. To achieve this the algorithm maintains the following data structures: We maintain an array $(p_1, p_2, \ldots, p_c)$ with the positions of the 1-bits in $x = (x_1, x_2, \ldots, x_n)$ where p_i, $1 \leq i \leq c$, is the position of the i-th 1-bit in x counted from the right (from the highest index n). Thus p_c is the position of the first 1-bit in x from the left (from the lowest index 1). It turns out that adding and removing 1's happens around the position p_c, so the length of this array changes dynamically. For $i > c$ the value of p_i is undefined (the algorithm does not 'clean up' those values after using them). We also maintain an array $(\nu_1, \nu_2, \ldots, \nu_c)$ to quickly find the smallest integer $j > i$ with $x_j = 0$ where i is the smallest integer with $x_i = 1$. However, the value of ν_i, $1 \leq i \leq c$, is defined only if p_i is a starting position of a substring of 1-bits in x; i.e., $x_{p_i} = 1$ and $x_{p_i - 1} = 0$, or $p_i = 1$. In this case, the value of ν_i is the index such that p_{ν_i} is the position where the corresponding substring of 1-bits ends in x. In particular, since p_c is the position of the first 1 in x, the value $p_{\nu_c} + 1$ is the smallest integer greater than p_c such that x has a 0-bit at this position. Initially, there is only one substring of 1-bits in $x = 1^c 0^{n-c}$ that starts at position $p_c = 1$ and ends at position $p_1 = c$, so we have $\nu_c = 1$ (see line T2). Assuming that the algorithm correctly computes the positions to flip the next bit(s) of x, it can be verified straightforwardly that it also correctly updates all relevant entries of p and ν in the four main cases (and their subcases).

To obtain a loopless algorithm that generates a tight enumeration of the vertices of $Q_{n,[k,l]}$ (instead of a saturating cycle), we simply call TrimGC$(n, k-1, l+1)$ and omit the first of the two Visit(x) calls in each of the lines T21, T24, T29, T34, T36 and T40. The algorithm then moves directly with a distance-2 step from a vertex x in level k or l to the vertex $s_{\Gamma_{n,k}}(x)$ or $s_{\Gamma_{n,l}}(x)$, respectively, without visiting down$(x, s_{\Gamma_{n,k}}(x))$ or up$(x, s_{\Gamma_{n,l}}(x))$ in between. Also, if $k = 0$ then line T2 has to be omitted, and the special case $c = 1$ and

$x = 0^{n-1}1$ must be treated separately in lines T17 and T18 (by setting $i := n - 1$). This completes the description of the algorithms for part (a) of Theorem 9. As mentioned before, the missing correctness proofs and the runtime analysis can be found in [14].

Acknowledgements. The authors thank Jiří Fink, Jerri Nummenpalo and Robert Šámal for several stimulating discussions about the problems discussed in this paper and the anonymous reviewers for their helpful comments.

References

1 URL: `http://www.math.tu-berlin.de/~muetze`.

2 J. Bitner, G. Ehrlich, and E. Reingold. Efficient generation of the binary reflected Gray code and its applications. *Comm. ACM*, 19(9):517–521, 1976.

3 M. Buck and D. Wiedemann. Gray codes with restricted density. *Discrete Math.*, 48(2-3):163–171, 1984. `doi:10.1016/0012-365X(84)90179-1`.

4 P. Chase. Combination generation and graylex ordering. *Congr. Numer.*, 69:215–242, 1989. Eighteenth Manitoba Conference on Numerical Mathematics and Computing (Winnipeg, MB, 1988).

5 P. Diaconis and R. Graham. *Magical mathematics*. Princeton University Press, Princeton, NJ, 2012. The mathematical ideas that animate great magic tricks, With a foreword by Martin Gardner.

6 D. Duffus, H. Kierstead, and H. Snevily. An explicit 1-factorization in the middle of the Boolean lattice. *J. Combin. Theory Ser. A*, 65(2):334–342, 1994. `doi:10.1016/0097-3165(94)90030-2`.

7 D. Duffus, B. Sands, and R. Woodrow. Lexicographic matchings cannot form Hamiltonian cycles. *Order*, 5(2):149–161, 1988. `doi:10.1007/BF00337620`.

8 P. Eades, M. Hickey, and R. Read. Some Hamilton paths and a minimal change algorithm. *J. Assoc. Comput. Mach.*, 31(1):19–29, 1984. `doi:10.1145/2422.322413`.

9 P. Eades and B. McKay. An algorithm for generating subsets of fixed size with a strong minimal change property. *Inform. Process. Lett.*, 19(3):131–133, 1984. `doi:10.1016/0020-0190(84)90091-7`.

10 G. Ehrlich. Loopless algorithms for generating permutations, combinations, and other combinatorial configurations. *J. Assoc. Comput. Mach.*, 20:500–513, 1973.

11 M. El-Hashash and A. Hassan. On the Hamiltonicity of two subgraphs of the hypercube. In *Proceedings of the Thirty-second Southeastern International Conference on Combinatorics, Graph Theory and Computing (Baton Rouge, LA, 2001)*, volume 148, pages 7–32, 2001.

12 S. Felsner and W. Trotter. Colorings of diagrams of interval orders and α-sequences of sets. *Discrete Math.*, 144(1-3):23–31, 1995. Combinatorics of ordered sets (Oberwolfach, 1991). `doi:10.1016/0012-365X(94)00283-O`.

13 F. Gray. Pulse code communication. March 17, 1953 (filed Nov. 1947). U.S. Patent 2,632,058.

14 P. Gregor and T. Mütze. Trimming and gluing Gray codes. *arXiv:1607.08806*, full version of this extended abstract, Jan 2017. URL: `https://arxiv.org/abs/1607.08806`.

15 P. Gregor and R. Škrekovski. On generalized middle-level problem. *Inform. Sci.*, 180(12):2448–2457, 2010. `doi:10.1016/j.ins.2010.02.009`.

16 I. Havel. Semipaths in directed cubes. In *Graphs and other combinatorial topics (Prague, 1982)*, volume 59 of *Teubner-Texte Math.*, pages 101–108. Teubner, Leipzig, 1983.

17 P. Horák, T. Kaiser, M. Rosenfeld, and Z. Ryjáček. The prism over the middle-levels graph is Hamiltonian. *Order*, 22(1):73–81, 2005. `doi:10.1007/s11083-005-9008-7`.

18 T. Jenkyns and D. McCarthy. Generating all k-subsets of $\{1 \cdots n\}$ with minimal changes. *Ars Combin.*, 40:153–159, 1995.

19 R. Johnson. Long cycles in the middle two layers of the discrete cube. *J. Combin. Theory Ser. A*, 105(2):255–271, 2004. doi:10.1016/j.jcta.2003.11.004.

20 H. Kierstead and W. Trotter. Explicit matchings in the middle levels of the Boolean lattice. *Order*, 5(2):163–171, 1988. doi:10.1007/BF00337621.

21 D. Knuth. *The Art of Computer Programming, Volume 4A*. Addison-Wesley, 2011.

22 S. Locke and R. Stong. Problem 10892: Spanning cycles in hypercubes. *Amer. Math. Monthly*, 110:440–441, 2003.

23 T. Mütze. Proof of the middle levels conjecture. *Proc. Lond. Math. Soc.*, 112(4):677–713, 2016. doi:10.1112/plms/pdw004.

24 T. Mütze and J. Nummenpalo. Efficient computation of middle levels Gray codes. In N. Bansal and I. Finocchi, editors, *Algorithms – ESA 2015*, volume 9294 of *Lecture Notes in Computer Science*, pages 915–927. Springer Berlin Heidelberg, 2015. *arXiv:1506.07898*. doi:10.1007/978-3-662-48350-3_76.

25 T. Mütze and J. Nummenpalo. A constant-time algorithm for middle levels Gray codes. *arXiv:1606.06172*. An extended abstract has been accepted for presentation at SODA 2017, June 2016.

26 T. Mütze and P. Su. Bipartite Kneser graphs are Hamiltonian. In J. Nešetril, O. Serra, and J. A. Telle, editors, *Electronic Notes in Discrete Mathematics – Eurocomb 2015*, volume 49, pages 259–267. Elsevier, 2015. *arXiv:1503.09175*. Accepted for publication in *Combinatorica*.

27 T. Mütze and F. Weber. Construction of 2-factors in the middle layer of the discrete cube. *J. Combin. Theory Ser. A*, 119(8):1832–1855, 2012. doi:10.1016/j.jcta.2012.06.005.

28 F. Ruskey. Adjacent interchange generation of combinations. *J. Algorithms*, 9(2):162–180, 1988. doi:10.1016/0196-6774(88)90036-3.

29 C. Savage. Long cycles in the middle two levels of the Boolean lattice. *Ars Combin.*, 35-A:97–108, 1993.

30 C. Savage. A survey of combinatorial Gray codes. *SIAM Rev.*, 39(4):605–629, 1997. doi:10.1137/S0036144595295272.

31 C. Savage and P. Winkler. Monotone Gray codes and the middle levels problem. *J. Combin. Theory Ser. A*, 70(2):230–248, 1995. doi:10.1016/0097-3165(95)90091-8.

32 I. Shields, B. Shields, and C. Savage. An update on the middle levels problem. *Discrete Math.*, 309(17):5271–5277, 2009. doi:10.1016/j.disc.2007.11.010.

33 M. Shimada and K. Amano. A note on the middle levels conjecture. *arXiv:0912.4564*, Sep 2011.

34 B. Stevens and A. Williams. The coolest way to generate binary strings. *Theory Comput. Syst.*, 54(4):551–577, 2014. doi:10.1007/s00224-013-9486-8.

35 D. Tang and C. Liu. Distance-2 cyclic chaining of constant-weight codes. *IEEE Trans. Computers*, C-22:176–180, 1973.

36 P. Winkler. *Mathematical puzzles: a connoisseur's collection.* A K Peters, Ltd., Natick, MA, 2004.