Designing q-Unique DNA Sequences with Integer Linear Programs and Euler Tours in De Bruijn Graphs

Marianna D'Addario[1,2], Nils Kriege[2], and Sven Rahmann[1,3]

1 **Collaborative Research Center (Sonderforschungsbereich, SFB) 876, TU Dortmund, Germany**
2 **Computer Science XI, TU Dortmund, Germany**
{Marianna.Daddario,Nils.Kriege}@tu-dortmund.de
3 **Genome Informatics, Institute of Human Genetics, Faculty of Medicine, University of Duisburg-Essen, Germany**
Sven.Rahmann@uni-due.de

Abstract

DNA nanoarchitechtures require carefully designed oligonucleotides with certain non-hybridization guarantees, which can be formalized as the q-uniqueness property on the sequence level. We study the optimization problem of finding a longest q-unique DNA sequence. We first present a convenient formulation as an integer linear program on the underlying De Bruijn graph that allows to flexibly incorporate a variety of constraints; solution times for practically relevant values of q are short. We then provide additional insights into the problem structure using the quotient graph of the De Bruijn graph with respect to the equivalence relation induced by reverse complementarity. Specifically, for odd q the quotient graph is Eulerian, so finding a longest q-unique sequence is equivalent to finding an Euler tour and solved in linear time with respect to the output string length. For even q, self-complementary edges complicate the problem, and the graph has to be Eulerized by deleting a minimum number of edges. Two sub-cases arise, for one of which we present a complete solution, while the other one remains open.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory, G.2.3 Applications

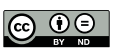
Keywords and phrases DNA sequence design, De Bruijn graph, quotient graph, reverse complement, Euler graph, Euler tour

Digital Object Identifier 10.4230/OASIcs.GCB.2012.82

1 Introduction

DNA synthesis technology allows to synthesize custom oligonucleotides of up to 80–100 basepairs (bp). Such oligonucleotides may assemble into larger regular structures, such as grids of 4x4 tiles [6]. These structures in turn provide a basis for attaching other molecules to protruding DNA arms, such as antibodies to capture specific proteins, which is why the research field of DNA nanoarchitechtures offers exciting future prospects for molecular biology and medicine.

Single-stranded DNA molecules form stable double-stranded helices according to the canonical Watson-Crick base pairing rules (A-T, G-C). The stability of a DNA double helix is given by a fine balance of interactions, such as hydrogen bonds between bases, stacking interactions of parallel bonds, and thermodynamic properties, such as the melting temperature or the free Gibbs Energy of a DNA sequence [9].



German Conference on Bioinformatics 2012 (GCB'12).
Editors: S. Böcker, F. Hufsky, K. Scheubert, J. Schleicher, S. Schuster; pp. 82–92

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The problem of designing appropriate DNA oligonucleotide sequences for assembly into a desired structure has been pioneered by Seeman, who in cooperation with Kallenbach developed the semi-automated SEQUIN software [10] for this purpose. Later, Feldkamp developed the DNA sequence compiler (`dsc` [7]) that allows to specify constraints between oligonucleotides as a mini-language and automates the search process by randomized construction and backtracking on sequences that violate the specifications.

For both approaches, a central aspect of the design problem is that the chosen oligonucleotides must be *q-unique* (see Section 2). Of course, besides q-uniqueness, other factors play an important role as well, such as uniform thermodynamic stability of all hybridizing regions in the desired structure. Here, however, we focus on the theoretical and combinatorial aspects of q-uniqueness.

We present a novel view on the computational problem of finding a longest q-unique DNA sequence. This is of practical relevance: A long q-unique sequence can be cut at arbitrary positions to obtain sets of oligonucleotides of any length that do not interact with each other. For practically relevant values of q, the problem is straightforwardly solved as an integer linear program (ILP) defined on a (standard directed) De Bruijn graph (Section 3). This approach has the advantage that it allows to incorporate quite general constraints (exclusion of certain substrings such as long homopolymers, inclusion of mandatory substrings, etc.) and can be solved with standard technology.

Interestingly, as we show in Section 4, the basic design problem can be cast as an Euler tour problem in an undirected variant of the De Bruijn graph, essentially the original De Bruijn graph modulo the equivalence relation induced by reverse complementarity. This graph is Eulerian for odd values of q, and the maximization problem has a closed-form solution. For even values of q, the graph is not Eulerian, and the problem arises to convert it into an Eulerian graph with a minimum number of edge (q-gram) deletions. An explicit solution is obtained when self-complementary edges are allowed in the graph, but when they are forbidden, a closed-form solution remains open.

2 Preliminaries

The *reverse complement* $\overline{s}$ of a sequence $s = \sigma_1\sigma_2\cdots\sigma_n$ over the DNA alphabet $\Sigma = \{\mathrm{A,C,T,G}\}$ is defined as $\overline{s} := \overline{\sigma}_n\overline{\sigma}_{n-1}\cdots\overline{\sigma}_1$, where $\overline{\mathrm{A}} = \mathrm{T}$, $\overline{\mathrm{T}} = \mathrm{A}$, $\overline{\mathrm{C}} = \mathrm{G}$ and $\overline{\mathrm{G}} = \mathrm{C}$. A length-$q$ string is an element of Σ^q and called a *q-gram*. A q-gram that is a substring a given sequence s is called a *q-gram of s*. In the following, for a given sequence s and a given (small) q, we consider the sequence of overlapping q-grams of s. For $s = \mathrm{GATTACA}$ and $q = 4$, we obtain the q-gram sequence (GATT, ATTA, TTAC, TACA).

A sequence s is said to be *q-unique* [7] if the following requirements are fulfilled.

1. Every q-gram occurs at most once in s.
2. If a q-gram occurs in s, then its reverse complement does not.

Note that the second requirement implies that (for even q) no self-complementary q-gram may occur in s. For odd q, self-complementary q-grams do not exist.

The q-uniqueness requirements (for small values of q) ensure that the designed sequence does not hybridize to itself in a stable way, except at exact reverse complementary counterparts, which are not part of the designed sequence, but introduced deliberately at a later stage. The first requirement ensures that each q-gram of s has exactly one reverse complementary counterpart and hence binds at a well-defined location to the reverse complement of s. If this requirement is violated, switching may occur, as shown in Figures 1a, 1b, which leads to unstable DNA structures. The second requirement ensures that s does not bind to itself; the

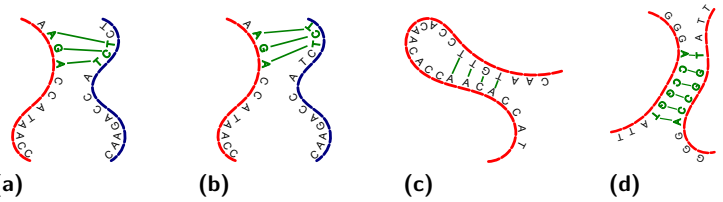


Figure 1 Sequences violating the q-uniqueness requirements. (a), (b): The q-gram TCT occurs twice, which leads to two alternative hybridizations. (c), (d): Existence of (self-)complementary q-grams results in the possibility of the sequence folding unto itself or binding to another molecule of the same type.

adverse case is shown in Figures 1c, 1d.

We note that there exist 4^q DNA q-grams. The number of self-complementary q-grams is zero if q is odd, and $4^{q/2}$ if q is even. Thus, an upper bound on the number of q-grams that can be used in a q-unique sequence is $M_q := 4^q/2$ if q is odd, and $M_q := (4^q - 4^{q/2})/2$ if q is even. If self-complementary q-grams are allowed (a variant of the problem that is irrelevant in practice, but combinatorially interesting), the M_q-bound for even q is increased by $4^{q/2}$. Thus $M'_q = 4^2/2$ if q is odd and $M'_q = (4^q + 4^{q/2})/2$ if q is even.

The optimization problem at hand is to find a q-unique sequence of maximal length. In the following, we study the question whether the upper bound M_q can be achieved. For this, the M_q selected q-grams would have to be ordered such that consecutive q-grams overlap by $(q-1)$ characters. We use a De Bruijn graph as a convenient data structure to study the optimization problem.

3 An Integer Linear Program on the De Bruijn Graph

We briefly recall the definition of a (directed) De Bruijn graph (DBG).

▶ **Definition 1** (De Bruijn graph)**.** The *De Bruijn graph* of order $q \geq 2$ over the alphabet Σ has vertices and edges

$$V := \Sigma^{q-1},$$
$$E := \{(\sigma_1 s, s\sigma_2) : \sigma_1, \sigma_2 \in \Sigma,\ s \in \Sigma^{q-2}\}.$$

Thus, the vertices are $(q-1)$-grams and the edges represent q-grams by connecting two vertices such that the source's $(q-2)$-suffix equals the sink's $(q-2)$-prefix.

Traversing a DBG using every edge at most once and avoiding edges labeled with self-complementary q-grams and complementary q-grams of used edges results in a path that describes a q-unique sequence. The optimization problem of finding a longest q-unique sequence can now be cast as an edge selection problem in the DBG: Find an edge sequence of maximal length that forms a path (or cycle), such that each edge is traversed at most once, traversal of an edge implies non-traversal of its reverse complement, and (for even q) no self-complementary edges are traversed.

This problem in turn can be conveniently cast as an integer linear program (ILP); see Table 1. Three sets of binary indicator variables are defined: $(x_e)_{e\in E}$, $(a_v)_{v\in V}$, and $(z_v)_{v\in V}$. The variable x_e indicates whether edge e is selected. Variables a_v and z_v indicate the start vertex and end vertex of a path, respectively. The objective function naturally maximizes the number of selected edges.

Table 1 ILP for selecting a maximal number of edges under q-uniqueness constraints. Selected edges may form cycles and/or one path. All variables are binary indicator variables: edge selection indicators x_e, path start indicators a_v, and path end indicators z_v. Functions γ, In and Out are defined in the text.

$$
\begin{aligned}
\text{Maximize } & \sum_{e \in E} x_e \\
\text{subject to } \quad & 0 \leq x_e \leq 1, \;\; x_e \in \mathbb{Z} && \forall e \in E, \\
& 0 \leq a_v \leq 1, \;\; a_v \in \mathbb{Z} && \forall v \in V, \\
& 0 \leq z_v \leq 1, \;\; z_v \in \mathbb{Z} && \forall v \in V, \\
& x_e + x_{\gamma(e)} \leq 1 && \forall e \in E, \\
& \sum_{e \in In(v)} x_e + a_v = \sum_{e \in Out(v)} x_e + z_v && \forall v \in V, \\
& \sum_{v} a_v \leq 1, \\
& \sum_{v} z_v \leq 1, \\
& a_v + z_v \leq 1 && \forall v \in V.
\end{aligned}
$$

Let $\gamma(e)$ denote the edge labeled with the reverse complement of e's label. To ensure q-uniqueness, we must not select both edge e and its complement, thus $x_e + x_{\gamma(e)} \leq 1$ for all edges $e \in E$. If $e = \gamma(e)$, this becomes $2x_e \leq 1$, which implies $x_e = 0$ because of the integer constraints.

Let $In(v)$ and $Out(v)$, respectively, be the set of incoming and outgoing edges of vertex v. To ensure the chosen edges form a path or cycle, we require that for every vertex the number of incoming edges equals the number of outgoing edges. If the path is not a cycle, the start vertex has one incoming edge less than outgoing edges, and the end vertex has one outgoing edge less than incoming ones. Thus we require $\sum_{e \in In(v)} x_e + a_v = \sum_{e \in Out(v)} x_e + z_v$ for all $v \in V$. To ensure that we obtain at most one start vertex and one end vertex and that they are not identical, we require $\sum_v a_v \leq 1$, $\sum_v z_v \leq 1$ and for all $v \in V : a_v + z_v \leq 1$.

Table 1 summarizes the ILP. Its conditions do not ensure that the solution is a single path. Instead, the solution may consist of one or more cycles and at most one path. To ensure a single path or cycle, further constraints may be added: Assume that the optimal solution of the ILP in Table 1 consists of two vertex-disjoint components, such as two cycles on vertex sets V' and V'', respectively. Then we add a constraint requiring the existence of an edge between V' and V'' and solve the ILP again. This technique of adding violated constraints on demand is called cutting-plane method. However, we found that the solution of the ILP from Table 1 already produced a single cycle or path, i.e., adding cutting planes was not necessary.

Table 2 shows the ILP's solution for $2 \leq q \leq 8$. The ILP was implemented in Python with the `python-zibopt` library [8] and solved with the SCIP solver [1]. The obtained solutions were single cycles for odd q and single paths for even q. The ILP's optimal solution agrees with the upper bound M_q for odd q (cf. Section 2), but differs for even $q \geq 4$. These findings suggest that for odd q, the problem is trivial and that the solution agrees with the upper bound. However, self-complementary q-grams appear to complicate the problem. We now present a different approach that proves that for odd q, the above conjecture holds, and that the solution can be obtained in linear time (without resorting to an ILP).

Table 2 ILP results on DNA De Bruijn graphs of order $2 \leq q \leq 8$. $|E_q|$: number of edges, $|E_q^{\text{sc}}|$: number of self-complementary edges, M_q: Upper bound on the number of selectable q-grams (see Section 2), opt: optimal ILP solution value ($\leq M_q$), diff: $M_q -$ opt.

q	$\lvert E_q\rvert$	$\lvert E_q^{\text{sc}}\rvert$	M_q	opt	diff
2	16	4	6	6	0
3	64	0	32	32	0
4	256	16	120	115	5
5	1024	0	512	512	0
6	4096	64	2016	1959	57
7	16384	0	8192	8192	0
8	65536	256	32640	32279	361

4 Euler Tours of the De Bruijn Quotient Graph

To represent the double-stranded nature of DNA directly in the graph, the central idea is to collapse vertices and edges whose label is the reverse complement of each other into a single vertex or edge. Formally, we define an equivalence relation $\equiv$ by $s \equiv t := (s = t \text{ or } s = \bar{t})$. Each equivalence class has size 1 (for self-complementary sequences) or 2 (all others). The De Bruijn graph modulo $\equiv$ on edges and vertices is called *De Bruijn Quotient Graph* (DBQG). It was introduced by Anderson et al. [2] in a different context (to construct universal DNA footprints). Each vertex of the quotient graph is jointly labeled with a $(q-1)$-gram and its reverse complement; each edge represents a q-gram and its reverse complement.

▶ **Definition 2** (De Bruijn quotient graph, DBQG)**.** The *De Bruijn quotient graph* of order $q \geq 2$ over the DNA alphabet Σ has vertices and edges

$$\tilde{V}_q := \{\{s, \overline{s}\} : s \in \Sigma^{q-1}\},$$
$$\tilde{E}_q := \left\{ \left\{ \{\sigma_1 s,\ \overline{s}\,\overline{\sigma_1}\}, \{s\sigma_2,\ \overline{\sigma_2}\,\overline{s}\} \right\} : \sigma_1, \sigma_2 \in \Sigma,\ s \in \Sigma^{q-2} \right\},$$

where $\overline{s}$ denotes the reverse complement of a sequence s.

We observe that in the DBQG of order q,

$$|\tilde{V}_q| = [4^{(q-1)} + 4^{(q-1)/2}]/2, \qquad |\tilde{E}_q| = 4^q/2 \qquad \text{if } q \text{ is odd},$$
$$|\tilde{V}_q| = 4^{(q-1)}/2, \qquad |\tilde{E}_q| = [4^q + 4^{q/2}]/2 \qquad \text{if } q \text{ is even}.$$

A key characteristic of the quotient graph is that it contains paths not corresponding to valid sequences. Although the edges of the quotient graph are undirected, choosing one of the edge labels determines a direction. As a consequence, a vertex v which is reached via an edge e_1 must be left via an edge e_2, such that the label of v, the $(q-1)$-suffix of a label of e_1 and the $(q-1)$-prefix of a label of e_2 are identical. Since in this case, we may also enter v via e_2 and leave via e_1, we refer to e_1 and e_2 as *admissible edge pair*. Consider the vertex {ACT, AGT} in Figure 2. By entering it via CACT, the vertex label ACT is selected. Now only the edges starting with ACT can be followed to leave the vertex, here {ACTC, GAGT}, {ACTG, CAGT}, {ACTT, AAGT}, or {ACTA, TAGT}.

An *admissible path* in the DBQG is a path whose internal vertices are entered and left via admissible edge pairs. We consider only admissible paths from now on and "path" always refers to admissible path.

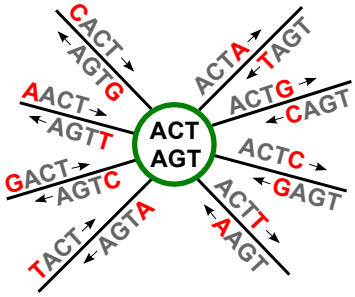
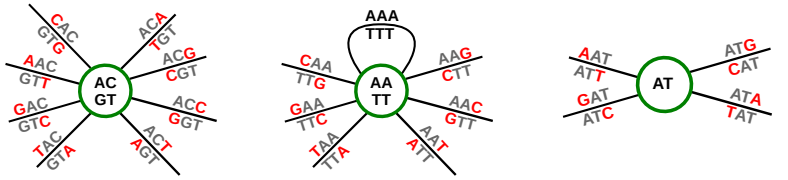


Figure 2 Admissible edge pairs (see text).



(a) Standard vertex **(b)** Homopolymer **(c)** Self-complementary

Figure 3 Vertex types of a DBQG with odd q.

The structure of the DBQG is more complex than the structure of the original DBG due to the presence of additional self-loops. In the DBG, an edge e is a self-loop from v to v if and only if e is labeled with a homopolymer, e.g., $\mathrm{AA}\ldots\mathrm{A} = \mathrm{A}^q$ and v is labeled with the corresponding homopolymer of length $q-1$. However, in the DBQG, an edge e is a self-loop if its label is either a homopolymer or self-complementary (this new case only occurs for even q). In the following sections, we precisely discuss this fact's implications on the graph structure and the maximization problem.

Each self-loop is part of two admissible edge pairs: First the entering edge and the loop are one admissible edge pair and then the loop and a leaving edge are the second one.

We define a vertex to be *balanced* if all incident edges can be arranged in admissible edge pairs, including self-loops. A DBQG is *Eulerian* if there exists a path or cycle that uses each edge exactly once. The following lemma is an adaptation of the same lemma for standard undirected graphs and proved in exactly the same way.

▶ **Lemma 3.** *A DBQG is Eulerian if and only if each vertex is balanced.*

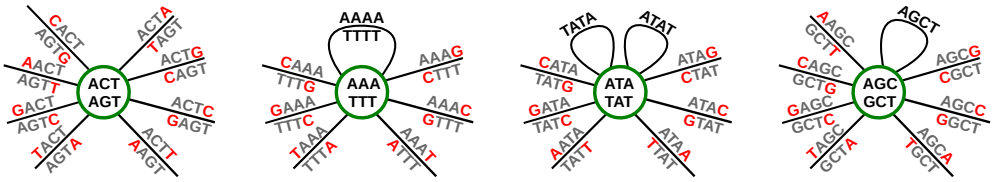
If the DBQG is Eulerian, an Euler tour may be constructed in linear time with a slight modification of the Hierholzer algorithm [4].

Note that a path that traverses each edge at most once and that does not traverse any self-complementary edge corresponds by definition to a q-unique double-stranded DNA sequence.

4.1 The Case of Odd q

Using the DBQG, we can understand why finding a longest q-unique sequence is easy when q is odd. In particular, the optimization problem can be solved in linear time with respect to the resulting q-unique sequence.

▶ **Theorem 4.** *For odd q, the DBQG is an Eulerian graph [2]. Each Euler tour is a cycle whose overlapping concatenated edge labels specify a q-unique sequence of maximum length.*



(a) Standard vertex **(b)** Homopolymer **(c)** Two self-complementary edges **(d)** One self-complementary edge

Figure 4 Vertex types of a DBQG with even q.

Proof. Recall from Section 1 that for odd q, the upper bound on the number of selectable q-grams is $M_q = 4^q/2 = |\tilde{E}_q|$. Thus if an Euler tour exists, each possible q-gram is used exactly once, and the upper bound is attained.

To prove that an Euler tour exists (this was already noted in [2]), we need to show that each vertex is balanced according to Lemma 3. Thus we consider the possible vertex types in a DBQG of odd order; see Figure 3. There are *self-complementary* vertices (the vertex label is self-complementary), *homopolymer* vertices (the vertex label is the repetition of a single nucleotide) and *standard* vertices. Note that the self-loop of a homopolymer vertex can be traversed when entering and leaving the vertex via an admissible edge pair. While the incident edges differ in number and type, each vertex is balanced. ◀

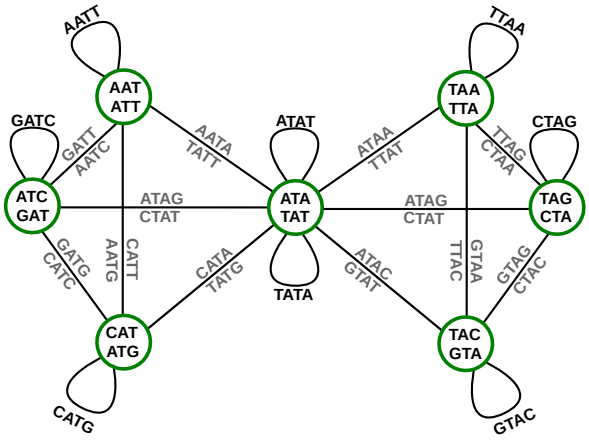
4.2 The Case of Even q

As in the odd case, we consider all types of vertices in a DBQG for even q, see Figure 4 and note that the graph is not Eulerian due to unbalanced vertices with one self-complementary edge (Figure 4d). It follows that an Euler tour cannot exist, and the bound M_q cannot be attained, explaining the differences in Table 2.

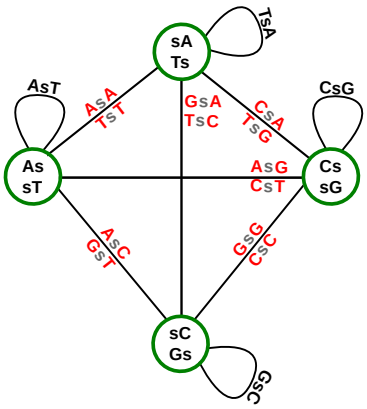
Finding a longest q-unique sequence now becomes the following problem: Given an undirected graph $G = (V, E)$, find an Eulerian graph $G^* = (V, E^*)$ with $E^* \subseteq E$ such that $|E \setminus E^*|$ is minimal. We call this process Eulerization-by-deletion. (This use of the term Eulerization is non-standard, as the literature reserves the term for Eulerization by duplication of existing edges.) In standard undirected graphs, Eulerization-by-deletion is polynomially solvable via minimum-weight perfect matchings [3, 5], where in our case the weight corresponds to the path length between the matched odd-degree vertices. Unfortunately this approach cannot be directly applied to the DBQG because of the restrictions imposed by admissible edge pairs.

We consider two cases for even q. The first one is of theoretical interest and assumes that the self-complementary edges still exist in the DBQG (i.e., self-complementary q-grams are allowed). We present a simple construction to optimally Eulerize the DBQG, so finding a maximal q-unique sequence is polynomially solvable. The second case is the original problem, where self-complementary edges are removed. For this case, we also present a construction to Eulerize the graph such that an Eulerian path exists; however, this construction is not optimal. Therefore, we only obtain a lower bound on the number of selectable edges, and the complexity remains open.

DBQG with self-complementary edges. Four types of vertices exist in the graph (Figure 4); only the vertices with one self-complementary edge (Figure 4d) are unbalanced. We investigate the the odd-degree vertices and their relations in the graph assuming $q \geq 4$.



(a) Bat-group: To obtain an Euler tour in the graph, four edges of this group must be deleted. Two of these must be incident to the central vertex. Every other vertex is incident to exactly one deleted edge. Non-deletable edges between these vertices are not shown.



(b) Kite-group: The substring s of length $q-2$ is self-complementary. To obtain an Euler tour in the graph, two edges of this group must be deleted and each vertex is incident to exactly one deleted edge.

Figure 5 Substructures in DBQG with even q.

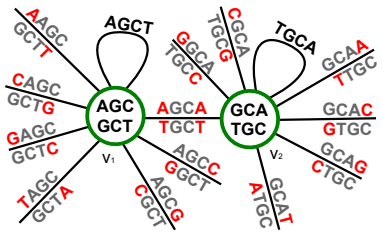


Figure 6 After deleting the central edge {AGCA, TGCT}, the vertices are balanced. Possible sets of paths through the vertex are {AAGC, GCTT}→{AGCC, GGCT}, {CAGC, GCTG}→{AGCG, CGCT} and {GAGC, GCTC}→{AGCT}→{GCTA, TAGC}. Using the self-complementary edge {AGCT} is essential for the balance.

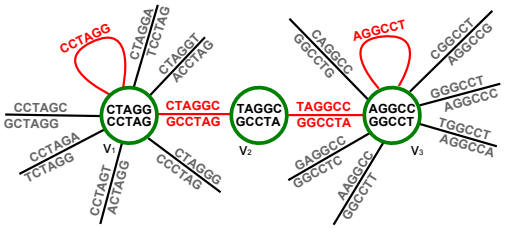


Figure 7 Path of length $q/2-1$ obtained by cyclic permutation of self-complementary edge CCTAGG into AGGCCT. When the self-complementary edges and the path are removed, all vertices on the path are balanced.

A *bat-group* is a set of 7 vertices centered around a vertex labeled $\{\sigma(\overline{\sigma}\sigma)^{q/2-1}, \overline{\sigma}(\sigma\overline{\sigma})^{q/2-1}\}$, $\sigma \in \Sigma$ (Figure 5a). There are only 2 bat-groups (the central vertex is labeled with ATAT... A or CGCG... C); each bat-group contains 8 self-complementary edges. The central vertex is balanced, but the other six are not. At least four edges must be deleted to balance each vertex in such a group, and Figure 5a shows several ways to achieve this.

Let s be a self-complementary $(q-2)$-gram. Then the *kite-group* of s is the set of 4 vertices that are not part of a bat-group and are labeled with $\{\sigma s, \overline{s}\,\overline{\sigma}\}$, for all $\sigma \in \Sigma$ (Figure 5b). Each kite-group contains four self-complementary edges. Because of the two bat-groups, there are $(4^{q/2}-16)/4$ kite-groups in the graph (for $q=4$, there are none). At least two edges must be deleted to balance each vertex in a kite-group, and Figure 5b shows all possibilities to achieve this. In more detail, Figure 6 shows that a single edge (the central edge in the figure) between two problematic vertices can be deleted, leaving both vertices balanced. For the balance, the self-complementary edge in each of the vertices is essential.

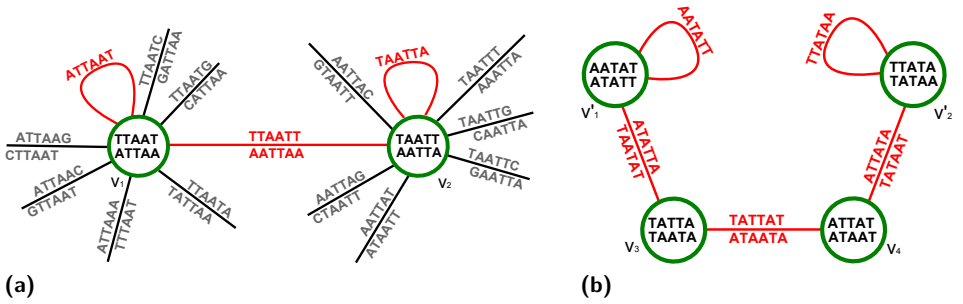


Figure 8 Alternative permutation path to delete: Vertex v_1 in (a) and vertex v_1' in (b) are the cyclic permutation pair described in Section 4; so are v_2 in (a) and v_2' in (b). Now, (a) shows a shorter path from v_1 to v_2 and (b) a longer one from v_1' to v_2'.

Summarizing, to Eulerize the graph, it is both necessary and possible to delete $\delta := 2 \cdot 4 + (4^{q/2} - 16)/4 \cdot 2 = 4^{q/2}/2$ edges. As we are equally satisfied with an Eulerian path instead of a cycle, we allow two unbalanced vertices and re-add two edges.

▶ **Theorem 5.** *In the DBQG for even q with self-complementary edges, an Eulerian subgraph with $N_q' := |\tilde{E}_q| - \delta + 2 = 4^q/2 + 2$ edges exists; larger Eulerian subgraphs do not exist.*

DBQG without self-complementary edges. When self-complementary edges are omitted, deleting an edge connecting two odd-degree vertices may leave them unbalanced (consider Figure 6 without the self-loops and with the central edge removed). Identifying an edge-minimal set of paths between pairs of these vertices, whose deletion leaves all vertices balanced, appears difficult and remains an open problem.

Here we present a systematic but sub-optimal construction. Consider a self-complementary q-gram of the form st, where s and t have length $q/2$ and are reverse complements of each other, and consider the vertices incident to st and its cyclic permutation ts. There is a path of length $q/2 - 1$ between these vertices (see Figure 7).

As there are two vertices with two self-complementary edges (Figure 4c), there are $4^{q/2}/2 - 2$ such (s, t) pairs. Since an Eulerian path instead of a cycle suffices, one of these paths may remain, resulting in $\Delta := (4^{q/2}/2 - 3)(q/2 - 1)$ edge deletions.

▶ **Theorem 6.** *In the DBQG for even q without self-complementary edges, an Eulerian subgraph with*

$$N_q := (|\tilde{E}_q| - 4^{q/2}) - \Delta = (4^q - 4^{q/2})/2 - (4^{q/2}/2 - 3)(q/2 - 1)$$

edges exists. Larger Eulerian subgraphs may exist.

For even $q \geq 6$, the value of N_q does not match the optimal ILP solution and is therefore suboptimal (e.g., $q = 6$: $N_q = 1958$, optimum 1959; $q = 8$: $N_q = 32265$, optimum 32279). For $q = 6$, we can manually improve the Eulerization to match the ILP solution: Due to the symmetry of some self-complementary edges it is possible to find an alternative shorter path between two self-complementary edges. Figure 8 shows this alternative pairing. Deleting only the path in Figure 8a and preserving that in Figure 8b saves exactly one edge. In general, the problem of optimal Eulerization in this case remains open.

5 Discussion and Conclusion

We presented two approaches to generate q-unique sequences, via an ILP and an Euler tour in a De Bruijn quotient graph.

For odd q, the DBQG is Eulerian and an adaptation of a standard algorithm for finding Euler tours yields an optimal algorithm. Additionally, the number of different longest q-unique sequences is related to the number of Euler tours in the DBQG, which can be computed by a small modification (accounting for admissible pairs instead of all edges) of the BEST Theorem (de Bruijn, van Aardenne-Ehrenfest, Smith and Tutte, [11]) for standard undirected graphs.

For even q, the DBQG turns out not to be Eulerian, and edges must be removed to Eulerize it. In the variant where self-complementary edges are allowed, we presented an optimal Eulerization by deleting edges, proving this problem variant to be as easy as the case of odd q. When self-complementary edges are forbidden, the cyclic permutation construction gives suboptimal results compared to the optimal ILP solution for even $q \geq 6$.

In practice, the design problem of q-unique sequences is generally restricted to $q \in \{3, 4, 5\}$, but augmented by additional constraints: Some substrings might be prescribed (existing DNA libraries that must be used) and others forbidden (homopolymers of certain length, q-grams with too extreme GC-content). The ILP formulation supports such constraints by fixing certain variables and provides reasonable performance for relevant values of q.

Respecting constraints with the approach based on the DBQG corresponds to the removal of certain edges. Clearly, conducting a thorough structural analysis for individual constraints is prohibitive. Therefore, this work poses the general problem of Eulerizing a given quotient graph by removing a minimum number of edges, such that each vertex has even degree and allows admissible edge pairings.

Acknowledgments. We gratefully acknowledge funding by the DFG Collaborative Research Center (Sonderforschungsbereich, SFB) 876 "Providing Information by Resource-Constrained Data Analysis" within project TB1 (`http://sfb876.tu-dortmund.de`). We thank C. Niemeyer, B. Sacca and J. Ksienczyk for fruitful discussions, and P. Mutzel and T. Schwentick for very helpful pointers to the Eulerization-by-deletion problem. We also thank the participants of Dagstuhl seminar 12291 "Structure Discovery in Biology: Motifs, Networks & Phylogenies" organized by A. Dress, L. Parida and A. Apostolico for their valuable comments.

References

1 T. Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

2 J.W. Anderson, K.R. Fox, and G.A. Niblo. A fast algorithm for the construction of universal footprinting templates in DNA. *Journal of mathematical biology*, 52(3):307–342, 2006.

3 W. Cook and A. Rohe. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing*, 11(2):138–148, 1999.

4 R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer Verlag, 2005.

5 J. Edmonds and E.L. Johnson. Matching, euler tours and the chinese postman. *Math. Programming*, 5:88–124, 1973.

6 U. Feldkamp and C.M. Niemeyer. Rational design of DNA nanoarchitectures. *Angewandte Chemie International Edition*, 45(12):1856–1876, 2006.

7 U. Feldkamp, H. Rauhe, and W. Banzhaf. Software tools for DNA sequence design. *Genetic Programming and Evolvable Machines*, 4(2):153–171, 2003.

8 Ryan J. O'Neil. Python interfaces to the ZIB Optimization Suite. `http://code.google.com/p/python-zibopt/`. Accessed June 2012.

9 J. SantaLucia Jr, H.T. Allawi, and P.A. Seneviratne. Improved Nearest-Neighbor Parameters for Predicting DNA Duplex Stability. *Biochemistry*, 35(11):3555–3562, 1996.

10 N.C. Seeman and N.R. Kallenbach. Design of immobile nucleic acid junctions. *Biophysical journal*, 44(2):201–209, 1983.

11 T. van Aardenne-Ehrenfest and N.G. de Bruijn. Circuits and trees in oriented linear graphs. *Simon Stevin*, 28:203–217, 1951.