On NP-Hardness of the Paired de Bruijn Sound Cycle Problem

Evgeny Kapun and Fedor Tsarev

St. Petersburg National Research University of Information Technologies, Mechanics and Optics
Genome Assembly Algorithms Laboratory
197101, Kronverksky pr., 49, St. Petersburg, Russia
tsarev@rain.ifmo.ru
http://genome.ifmo.ru/

Abstract. The paired de Bruijn graph is an extension of de Bruijn graph incorporating mate pair information for genome assembly proposed by Mevdedev et al. However, unlike in an ordinary de Bruijn graph, not every path or cycle in a paired de Bruijn graph will spell a string, because there is an additional soundness constraint on the path. In this paper we show that the problem of checking if there is a sound cycle in a paired de Bruijn graph is NP-hard in general case. We also explore some of its special cases, as well as a modified version where the cycle must also pass through every edge.

Keywords: paired de Bruijn graph, genome assembly, complexity, NP-hard.

1 Introduction

Current genome sequencing technologies rely on the shotgun method — the genome is split into several small fragments which are read directly. Some of the technologies generate single reads, while others generate mate-pair reads — genome fragments are read from both sides. The problem of reconstructing the initial genome from these small fragments (reads) is known as the genome assembly problem. It is one of the fundamental problems of bioinformatics. Several models for genome assembly were studied by researchers.

One of the models for the single reads case is based on the maximum parsimony principle — the original genome should be the shortest string containing all reads as substrings. This leads to the Shortest Common Superstring (SCS) problem which is NP-hard [1]. In the de Bruijn graph model proposed in [8] each read is represented by a walk in the graph. Any walk containing all the reads as subwalks represents a valid assembly. Consequently, the genome assembly problem is formulated as finding the shortest superwalk. This problem, known as Shortest De Bruijn Superwalk problem (SDBS), was shown to be NP-hard [6].

In [5] an algorithm for reads' copy counts estimation based on maximum likelihood principle was proposed. A similar algorithm can be applied to find multiplicities of the de Bruijn graph edges, so, the De Bruijn Superwalk with Multiplicities problem (DBSM) can be formulated. This problem have been proven to be NP-hard as well [2].

Paired-end reads case is much less studied. To the best of our knowledge the only model which deals with paired-end reads is the paired de Bruijn graph proposed in [7]. However, not every path or cycle in a paired de Bruijn graph corresponds to a correct genome assembly, because there is an additional soundness constraint on the walk. Computational complexity for the problem of finding a sound cycle in the paired de Bruijn graph remained unknown [9]. In this paper we show that this problem is NP-hard.

2 Definitions

A *de Bruijn graph* of order k over an alphabet Σ is a directed graph in which every vertex has an associated label (a string over Σ) of length k and every edge has an associated label of length $k+1$. All labels within a graph must be distinct. If an edge (u, v) has an associated label l, then the label associated with u must be a prefix of l and the label associated with v must be a suffix of l.

Every path in a de Bruijn graph spells a string. A string spelled by a path v_1, e_1, v_2, $\ldots$, e_{n-1}, v_n of length n is a unique string s of length $n+k-1$ such that the label associated with v_i occurs in s at position i for all $1 \leq i \leq n$, and the label associated with e_i occurs in s at position i for all $1 \leq i \leq n-1$. Every cycle of length n in a de Bruijn graph spells a cyclic string of length n having the same properties.

In a *paired de Bruijn graph* each vertex and each edge has an associated *bilabel* instead of a label. A bilabel is an ordered pair of strings of the same length (equal to the order of the graph), denoted as (a, b). We say that (a_1, b_1) is a prefix of (a_2, b_2) iff a_1 is a prefix of a_2 and b_1 is a prefix of b_2. Suffix is defined analogously. As in ordinary de Bruijn graphs, all bilabels must be distinct, however, individual labels of which bilabels consist may coincide.

Similarly to the ordinary de Bruijn graph, every path in a paired de Bruijn graph spells a pair of strings, and every cycle spells a pair of cyclic strings. We say that a pair of strings $(s_1 s_2 \ldots s_n, t_1 t_2 \ldots t_n)$ of length n *matches with shift* d iff $s_{i+d} = t_i$ for all $1 \leq i \leq n-d$. Analogously, a pair of cyclic strings $(s_1 s_2 \ldots s_n, t_1 t_2 \ldots t_n)$ matches with shift d iff $s_{i+d} = t_i$ for all $1 \leq i \leq n-d$ and $s_i = t_{i+n-d}$ for all $1 \leq i \leq d$.

We say that a path in a paired de Bruijn graph is *sound with respect to shift* d, or just *sound*, iff the pair of strings it spells matches with shift d. We say that a cycle in a paired de Bruijn graph is sound iff the pair of cyclic strings matches with shift d.

We say that a path or a cycle is *covering* if it includes all the edges in a graph. We say that a set of paths or cycles covers the graph iff every edge of the graph belongs to at least one path or cycle in the set.

A *promise problem* is a kind of decision problem where only inputs from some set of valid inputs are considered. Specifically, a promise problem is defined by a pair of disjoint sets (S_+, S_-). A solution to the problem is a program which outputs "yes" when run on inputs in S_+ and outputs "no" when run on inputs in S_-. However, when run on inputs outside of $S_+ \cup S_-$, its behavior may be arbitrary: it may return any result, exceed its allowed time and memory bounds, or even hang.

Note that a promise problem (S_+, S_-) is at most as hard as (S'_+, S'_-) if $S_+ \subseteq S'_+$ and $S_- \subseteq S'_-$, because the solution for the latter problem would solve the former problem as well. Particularly, (S'_+, S'_-) is NP-hard if (S_+, S_-) is NP-hard. Also, an ordinary decision problem defined by set S is the same as the promise problem $(S, \complement S)$ (here, $\complement$ means set complement).

In the following problems, it would be assumed that the input consists of Σ, an alphabet, G, a paired de Bruijn graph of order k over Σ, as well as 1^d, that is unary coding of d.

3 Trivial Cases

If $|\Sigma| = 1$, a paired de Bruijn graph can have at most one vertex and at most one edge, and every cycle is sound. If $k = 0$, a paired de Bruijn graph can have at most one vertex and at most $|\Sigma|^2$ edges, and the problem is a bit harder. However, it can be solved in polymonial time in the following way: construct a directed graph with one vertex for each element of Σ and edge (u, v) iff there is an edge labeled with (u, v) in the original graph (this new graph may contain loops). Now, there is a sound cycle in the original graph iff there is a cycle in the new graph, and there is a covering sound cycle in the original graph iff there is a set of at most d cycles covering the new graph. Both properties can be easily checked in polymonial time.

4 A Case with Fixed *k*

Theorem 1. *For any fixed $k \geq 1$, the promise problem (S_+, S_-), where S_+ is the set of paired de Bruijn graphs which have a covering sound cycle and S_- is the set of paired de Bruijn graphs which do not have a sound cycle, is NP-hard.*

Proof. The proof of this theorem consists of two parts. Firstly, NP-hardness of a specific graph theory problem is proven by reduction from Hamiltonian Cycle problem. Then, the intermediate problem is reduced to the problem formulated in the theorem.

Lemma 1. *The promise problem (S_+, S_-), where S_+ is the set of undirected graphs with a hamiltonian cycle and S_- is the set of undirected graphs without hamiltonian paths, is NP-hard.*

Proof. First note that the problem is well-defined, because every graph with a hamiltonian cycle has a hamiltonian path. We will start with an instance G of Hamiltonian Cycle problem, which is NP-hard [3]. Without loss of generality, let us assume that G has at least three vertices.

Now build a new graph G' in the following way: firstly, pick a vertex in G and duplicate it together with all the edges incident to it. Let the copies of the vertex be a_1 and b_1. Now let us duplicate the whole graph, let the first copy be G_1 and the second copy be G_2, and let the copies of a_1 and b_1 be a_3 and b_3. Add two new vertices a_2 and b_2 and four new edges $\{a_1, a_2\}$, $\{a_2, a_3\}$, $\{b_1, b_2\}$, and $\{b_2, b_3\}$ (see Figure 1).

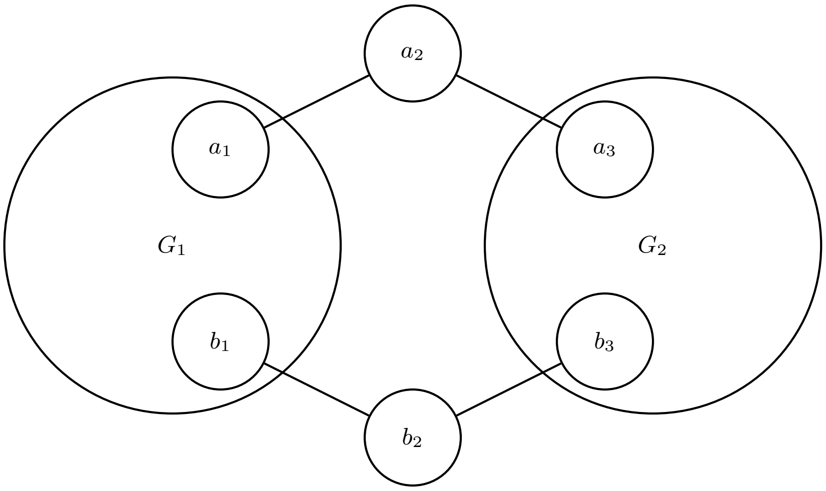

Fig. 1. Graph G'

The following two theorems show that the transformation described above maps all positive instances of hamiltonian cycle problem to S_+ and all negative instances of hamiltonian cycle problem to S_-.

Theorem 2. *If a graph G has a hamiltonian cycle, then the graph G' produced as described above has a hamiltonian cycle.*

Proof. After the vertex in G is duplicated, the cycle in G maps to a hamiltonian path in G_1 from a_1 to b_1. Analogously, G_2 has a hamiltonian path from a_3 to b_3. So, the cycle in G' is constructed as follows: start at a_1, traverse the path in G_1 to b_1, go to b_2, then to b_3, then traverse the path in G_2 to a_3, then go to a_2, and return to a_1.

Theorem 3. *If a graph G does not have hamiltonian cycles, then the graph G' does not have hamiltonian paths.*

Proof. Suppose that, on the contrary, G' contains a hamiltonian path. First consider the case when one end of the path is in G_1 and the other end is in G_2. Then, the path either traverses edges $\{a_1, a_2\}$ and $\{a_2, a_3\}$ but not $\{b_1, b_2\}$ and $\{b_2, b_3\}$, or $\{b_1, b_2\}$ and $\{b_2, b_3\}$ but not $\{a_1, a_2\}$ and $\{a_2, a_3\}$. In both cases, either a_2 or b_2 is not visited, so the path is not hamiltonian. So, ends of the path are either both outside G_1, or both outside G_2. Let us assume they are outside G_1, the other case is proved analogously. Besides a_1 and b_1, G_1 contains at least one internal vertex because of the assumption that G has at least three vertices. To reach that vertex, the path must enter G_1 through a_1 and leave through b_1 (or the opposite, which doesn't matter). Because there are no other ways to enter G_1, the path enters G_1 only once and traverses all vertices of G_1. So, the fragment of the path within G_1, when mapped back to G, becomes a hamiltonian cycle. So, G has a hamiltonian cycle, a contradiction.

Lemma 2. *If a graph has a hamiltonian cycle, then:*

- *For each vertex v in the graph, there is a hamiltonian path having v as one of its endpoints.*
- *For each edge $\{u, v\}$ in the graph, there is a hamiltonian path passing through $\{u, v\}$.*
- *For each edge $\{u, v\}$ and vertex $w \neq u, v$, there is a hamiltonian path passing through $\{u, v\}$ such that v resides between u and w on the path.*

Proof. Let n be the number of vertices in the graph, and let $v_1, v_2, \ldots, v_n$ be the vertices numbered in the order of the cycle. Let $u = v_i$ and $v = v_j$, $i < j$ (otherwise, vertices can be renumbered in the reverse order), and let $w = v_k$. Then, the first point of the theorem is obvious, the path for the second point is $v_{j+1}, v_{j+2}, \ldots, v_n, v_1, v_2, \ldots, v_i, v_j, v_{j-1}, \ldots, v_{i+1}$, and the path for the third point is the same if $i < k < j$ and $v_{j-1}, v_{j-2}, \ldots, v_i, v_j, v_{j+1}, \ldots, v_n, v_1, v_2, \ldots, v_{i-1}$ otherwise.

Now return to Theorem 1. First consider the case $k = 1$. Begin with an instance G of the problem from Lemma 1. Let G have n vertices $v_1, v_2, \ldots, v_n$. Without loss of generality, let us assume that $n \geq 3$. Set $d = n + 1$. Now, we are going to construct a paired de Bruijn graph $G' = (V, A)$. It would have block structure: there will be $2n + 2$ blocks $V_1, V_2, \ldots, V_{2n+2}$ and $2n + 2$ separator vertices $s_1, s_2, \ldots, s_{2n+2}$, so $V = V_1 \cup V_2 \cup \cdots \cup V_{2n+2} \cup \{s_1, s_2, \ldots, s_{2n+2}\}$ (see Figure 2). This graph will contain edges of three kinds:

- Within a block.
- From s_i to an element of V_i.
- From an element of V_i to s_{i+1}, or from an element of V_{2n+2} to s_1.

The alphabet would be analogously divided into $2n + 2$ blocks $C_1, C_2, \ldots, C_{2n+2}$ and $2n + 2$ separator characters $t_1, t_2, \ldots, t_{2n+2}$, so $\Sigma = C_1 \cup C_2 \cup \cdots \cup C_{2n+2} \cup \{t_1, t_2, \ldots, t_{2n+2}\}$. For each vertex $v \in V_i$, the first component of the associated bilabel will be in C_i, and the second component will be in C_{i+1} (or C_1 if $i = 2n + 2$). Each s_i would be associated with a bilabel (t_i, t_{i+1}), s_{2n+2} will be associated with a bilabel (t_{2n+2}, t_1).

The blocks will be formed as follows: the blocks V_1 and V_{2n+2} would be copies of G, while blocks V_2 through V_{2n+1} would each contain two copies of G, except for one vertex of which only one copy would be present. The vertices from the

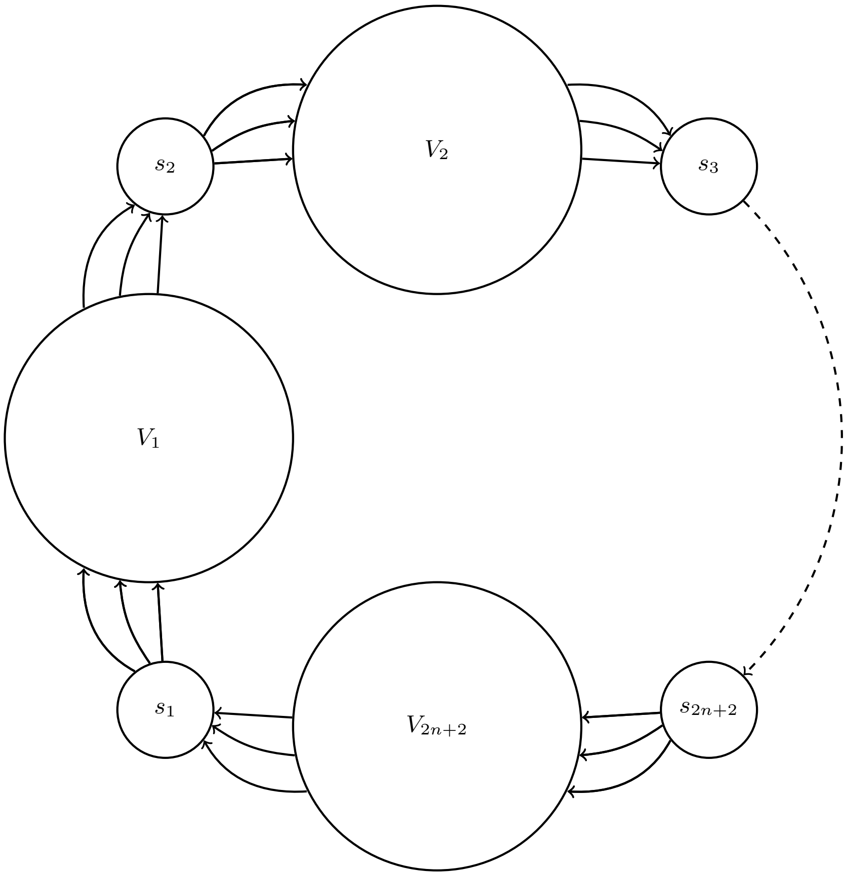

Fig. 2. Structure of the paired de Bruijn graph

first copy would be called $v_{i,j}$, the vertices from the second copy would be called $v''_{i,j}$, and the only copy of $v_{\lfloor i/2 \rfloor}$ in block i would be called $v'_{i,\lfloor i/2 \rfloor}$. The edges would be added such that every path through such block would pass through this vertex.

By assigning a dedicated subset of the alphabet to each block, we prevent vertices from different blocks from being assigned the same bilabel. In fact, it can be seen from the following definition that each vertex is assigned a distinct bilabel, so the assignment is valid. We also note that, with the exceptions of the bilabels containing u, the second index of a character (j in $c_{i,j}$) is the same in both components of a bilabel. This means that in a sound path the sequence of second indices must repeat with a period of d.

The precise definition is as follows:

- For $i = 1, 2n+2$ block $V_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,n}\}$.
- For $i = 2 \ldots 2n+1$ block $V_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,\lfloor i/2 \rfloor -1}, v_{i,\lfloor i/2 \rfloor +1}, v_{i,\lfloor i/2 \rfloor +2}, \ldots, v_{i,n}, v'_{i,\lfloor i/2 \rfloor}, v''_{i,1}, v''_{i,2}, v''_{i,\lfloor i/2 \rfloor -1}, v''_{i,\lfloor i/2 \rfloor +1}, v''_{i,\lfloor i/2 \rfloor +2}, \ldots, v''_{i,n}\}$.

- Alphabet block $C_1 = \{u\}$.
- For $i = 1 \dots n+1$ alphabet block $C_{2i} = \{c_{2i,1}, c_{2i,2}, \dots, c_{2i,n}\}$.
- For $i = 1 \dots n$ alphabet block $C_{2i+1} = \{c_{2i+1,1}, c_{2i+1,2}, \dots, c_{2i+1,i-1}, c_{2i+1,i+1}, c_{2i+1,i+2}, \dots, c_{2i+1,n}, c'_{2i+1,i}, c''_{2i+1,1}, c''_{2i+1,2}, \dots, c''_{2i+1,i-1}, c''_{2i+1,i+1}, c''_{2i+1,i+2}, \dots, c''_{2i+1,n}\}$.
- For $i = 1 \dots n$ the bilabel associated with $v_{1,i}$ is $(u, c_{2,i})$.
- For $i = 1 \dots n$ the bilabel associated with $v_{2n+2,i}$ is $(c_{2n+2,i}, u)$.
- For $i = 2 \dots 2n+1$, $j = 1 \dots n$, $j \neq \lfloor i/2 \rfloor$ the bilabel associated with $v_{i,j}$ is $(c_{i,j}, c_{i+1,j})$.
- For $i = 1 \dots n$ the bilabel associated with $v'_{2i,i}$ is $(c_{2i,i}, c'_{2i+1,i})$.
- For $i = 1 \dots n$ the bilabel associated with $v'_{2i+1,i}$ is $(c'_{2i+1,i}, c_{2i+2,i})$.
- For $i = 1 \dots n$, $j = 1 \dots n$, $j \neq i$ the bilabel associated with $v''_{2i,j}$ is $(c_{2i,j}, c''_{2i+1,j})$.
- For $i = 1 \dots n$, $j = 1 \dots n$, $j \neq i$ the bilabel associated with $v''_{2i+1,j}$ is $(c''_{2i+1,j}, c_{2i+2,j})$.

The edges are added:

- For $i = 1 \dots n$ the edges $(s_1, v_{1,i})$, $(v_{1,i}, s_2)$, $(s_{2n+2}, v_{2n+2,i})$, and $(v_{2n+2,i}, s_1)$.
- For $i = 2 \dots 2n+1$, $j = 1 \dots n$, $j \neq \lfloor i/2 \rfloor$ the edges $(s_i, v_{i,j})$ and $(v''_{i,j}, s_{i+1})$.
- For $i = 2 \dots 2n+1$ the edges $(s_i, v'_{i,\lfloor i/2 \rfloor})$ and $(v'_{i,\lfloor i/2 \rfloor}, s_{i+1})$.

Also, for each edge $\{v_i, v_j\}$ in G ($1 \leq i \leq n$, $1 \leq j \leq n$, $i \neq j$) the following edges are added:

- The edges $(v_{1,i}, v_{1,j})$, $(v_{2n+2,i}, v_{2n+2,j})$, $(v_{2j,i}, v'_{2j,j})$, $(v_{2j+1,i}, v'_{2j+1,j})$, $(v'_{2i,i}, v''_{2i,j})$, and $(v'_{2i+1,i}, v''_{2i+1,j})$.
- For $r = 2 \dots 2n+1$, $i \neq \lfloor r/2 \rfloor$, $j \neq \lfloor r/2 \rfloor$ the edges $(v_{r,i}, v_{r,j})$ and $(v''_{r,i}, v''_{r,j})$.

Note that, as edges of G are undirected, each edge should be processed twice, once as $\{v_i, v_j\}$ and once as $\{v_j, v_i\}$. The bilabels associated with the edges can be unambiguously determined from the bilabels associated with their ends.

The size of G' and the parameter d is polymonial in terms of n by construction. The following two theorems show that the transformation described above maps all positive instances of the problem formulated in Lemma 1 to paired de Bruijn graphs with covering sound cycles and all negative instances of the problem formulated in Lemma 1 to paired de Bruijn graphs without sound cycles.

Theorem 4. *If a graph G has a hamiltonian cycle, then the paired de Bruijn graph G' produced as described above has a covering sound cycle.*

Proof. Remember that $d = n + 1$. Construct the cycle as follows: first, select a hamiltonian path in G, let it be v_{p_1}, v_{p_2}, …, v_{p_n}. Start at s_1, then go to v_{1,p_1}, v_{1,p_2}, …, v_{1,p_n}. Then, for each i from 2 to $2n+1$, visit s_i, then v_{i,p_1}, v_{i,p_2}, …, $v_{i,p_{r_{\lfloor i/2 \rfloor}-1}}$, where $r_{\lfloor i/2 \rfloor}$ is such that $p_{r_{\lfloor i/2 \rfloor}} = \lfloor i/2 \rfloor$, then $v'_{i,p_{r_{\lfloor i/2 \rfloor}}} = v'_{i,\lfloor i/2 \rfloor}$, then $v''_{i,p_{r_{\lfloor i/2 \rfloor}+1}}$, $v''_{i,p_{r_{\lfloor i/2 \rfloor}+2}}$, …, v''_{i,p_n}. After that, visit s_{2n+2}, v_{2n+2,p_1}, v_{2n+2,p_2}, …, v_{2n+2,p_n}, and finally return to s_1.

This cycle visits each block, and the sequence of second indices within each block is the same (it is p_1, p_2, …, p_n), therefore, from the construction, the cycle is sound. However, it is not necessarily covering. To make a covering cycle, first use the procedure described above to construct one cycle per every property from Lemma 2, namely:

– For every vertex v_i, use the path having v_i as an endpoint to construct cycles passing through $(s_j, v_{j,i})$ $(1 \le j \le 2n+2,\ i \ne \lfloor j/2 \rfloor)$, $(s_{2i}, v'_{2i,i})$, $(s_{2i+1}, v'_{2i+1,i})$, $(v_{1,i}, s_2)$, $(v_{2n+2,i}, s_1)$, $(v'_{2i,i}, s_{2i+1})$, $(v'_{2i+1,i}, s_{2i+2})$, and $(v''_{j,i}, s_{j+1})$ $(2 \le j \le 2n+1,\ i \ne \lfloor j/2 \rfloor)$.
– For every edge $\{v_i, v_j\}$, use the path passing through $\{v_i, v_j\}$ to construct cycles passing through $(v_{r,i}, v_{r,j})$ $(r = 1, 2n+2)$, $(v_{2j,i}, v'_{2j,j})$, $(v_{2j+1,i}, v'_{2j+1,j})$, $(v'_{2i,i}, v''_{2i,j})$, and $(v'_{2i+1,i}, v''_{2i+1,j})$.
– For every edge $\{v_i, v_j\}$ and vertex v_k $(k \ne i, j)$, use the path passing through $\{v_i, v_j\}$, such that v_j resides between v_i and v_k on the path, to construct cycles passing through $(v_{2k,i}, v_{2k,j})$, $(v_{2k+1,i}, v_{2k+1,j})$, $(v''_{2k,j}, v''_{2k,i})$, and $(v''_{2k+1,j}, v''_{2k+1,i})$.

Together, these cycles should cover all the edges of G'. To make a single covering cycle, cut all these cycles at s_1 and join them together. The resulting cycle is sound because the second component of every bilabel from V_{2n+2} is u, and the first component of every bilabel from V_1 is also u, so they always match.

Theorem 5. *If a graph G doesn't have hamiltonian paths, then the paired de Bruijn graph G' produced as described above doesn't have sound cycles.*

Proof. Within each block, the set of characters used for the first component of bilabels and the set of characters used for the second component of the bilabel do not intersect. Therefore, every contiguous segment of a sound cycle within a single block must have length at most d. Because the blocks are connected in a circle (see Figure 2), and the cycle cannot be contained within a single block, it must pass around the circle at least once. Therefore, it must pass through s_1. Exactly d vertices later, it must pass through s_2, as it is the only vertex with a matching bilabel. The $d - 1 = n$ vertices between s_1 and s_2 must be spent within V_1, as the only other way to get to s_2 is to pass around the whole circle at least once, and the circle is longer than d, so this is impossible. Then it must pass through s_3, V_3, s_4, V_4, …, s_{2n+2}, V_{2n+2}, then return to s_1.

Let us call a segment between successive visits to s_1 a pass. Within a pass, each block is visited exactly once, and a path within each block has length n. Moreover, every pair of consecutive blocks, except (V_{2n+2}, V_1), has their vertices labeled such that the sequences of second indices within each block must be the same. However, for each i, such that $1 \le i \le n$, the structure of blocks V_{2i} and V_{2i+1} requires the sequence of second indices to include i, as it it impossible to pass though these blocks otherwise. Therefore, the sequence must include every value from 1 to n, so it is a permutation. Since every edge in G' within a block corresponde to an edge in G, the permutation defines a hamiltonian path in G, a contradiction.

The case $k > 1$ is handled as follows: first, produce a graph G over an alphabet Σ for the case $k = 1$. Then, construct a new alphabet Σ' as being equal to $\Sigma \cup \{f\}$, where f is a new character. After that, construct a new graph G' from G by replacing each vertex labeled (a, b) with k' vertices labeled $(f^{k'-1}a, f^{k'-1}b)$, $(f^{k'-2}af, f^{k'-2}bf)$, ..., $(af^{k'-1}, bf^{k'-1})$ and $k'-1$ edges labeled $(f^{k'-1}af, f^{k'-1}bf)$, $(f^{k'-2}aff, f^{k'-2}bff)$, ..., $(faf^{k'-1}, fbf^{k'-1})$, and replacing each edge labeled with (ab, cd) with an edge labeled $(af^{k'-1}b, cf^{k'-1}d)$. Finally, set d' equal $k'd$. Now, every sound cycle in can be unambiguously mapped from G to G' and vice versa. Therefore, the new solution is equivalent to the old one.

These immediately follow from Theorem 1:

Corollary 1. *The problem of checking whether a paired de Bruijn graph contains a sound cycle is NP-hard, both in general case and for any fixed* $k \geq 1$.

Corollary 2. *The problem of checking whether a paired de Bruijn graph contains a covering sound cycle is NP-hard, both in general case and for any fixed* $k \geq 1$.

5 A Case with Fixed $|\Sigma|$

Theorem 6. *For any fixed* $|\Sigma| \geq 2$*, the promise problem* (S_+, S_-)*, where* S_+ *is the set of paired de Bruijn graphs which have a covering sound cycle and* S_- *is the set of paired de Bruijn graphs which don't have a sound cycle, is NP-hard.*

Proof. This is proven by reduction from the same problem with fixed $k = 1$. Let the instance with $k = 1$ be G, and let its alphabet be Σ. We are going to build an instance G' of the same problem with alphabet $\Sigma' = \{0, 1\}$. Set $l = \lceil \log_2 |\Sigma| \rceil$. Now, every character from Σ can be unambiguously encoded with l binary digits. Take that encoding, and replace each digit 0 with the sequence 01, and each digit 1 with the sequence 10. The resulting encoding of length $2l$ has the following properties: it does not contain repetitions of three or more of the same digit as a substring, and it does not begin or end with a repeated digit. Set $k' = 4l + 5$. Let $\mathrm{enc}(c)$ denote the $2l$-character encoding of c described above. Then, for each vertex in G labeled (a, b), add a vertex labeled $(\mathrm{enc}(a)01110\,\mathrm{enc}(a), \mathrm{enc}(b)01110\,\mathrm{enc}(b))$. Here, the sequence 111 unambiguously determines the center of the encoding of a character. Each edge from G is translated to $4l + 9$ new vertices and $4l + 10$ new edges: if the original edge has the bilabel (ab, cd), the bilabels of the new vertices and edges will spell $(\mathrm{enc}(a)01110\,\mathrm{enc}(a)10001\,\mathrm{enc}(b)01110\,\mathrm{enc}(b)$, $\mathrm{enc}(c)01110\,\mathrm{enc}(c)10001\,\mathrm{enc}(d)01110\,\mathrm{enc}(d))$. Each bilabel would include at least one of the marker sequences 000 and 111 and at least one complete encoding of a character, so there will be no undesired overlaps. It can be shown that each sound cycle from G can be mapped to G' and vice versa, so they are equivalent for the purposes of the problem.

These immediately follow from Theorem 6:

Corollary 3. *The problem of checking whether a paired de Bruijn graph contains a sound cycle is NP-hard for any fixed $|\Sigma| \geq 2$.*

Corollary 4. *The problem of checking whether a paired de Bruijn graph contains a covering sound cycle is NP-hard for any fixed $|\Sigma| \geq 2$.*

6 A Case with Both k and $|\Sigma|$ Fixed

If both k and $|\Sigma|$ are fixed, the number of possible paired de Bruijn graphs is limited: there are at most $|\Sigma|^{2k}$ different vertex bilabels, and at most $|\Sigma|^{2k+2}$ different edge bilabels, and each bilabel is used by at most one vertex or edge, so the total number of different paired de Bruijn graphs is limited by a number which only depends on k and $|\Sigma|$. Let us denote this number by N.

There are at most N different problem instances for each instance length: otherwise, there would be two different instances having the same graph and the same length, but such instances can only differ in d, which is represented in unary coding, so any instances which only differ in d must have different length. Therefore, the number of instances is polynomial in instance length, so the language defined by the problem is *sparse*. Unless P=NP, a sparse language is never NP-hard [4]. Therefore, the problem of checking whether a paired de Bruijn graph has a sound cycle cannot be NP-hard if both k and $|\Sigma|$ are fixed.

7 Conclusion

We have proved that the Paired de Bruijn Sound Cycle problem is NP-hard in general case. Results of this work combined with previous works on genome assembly complexity show that all known models for genome assembly both from single and mate-pair reads are NP-hard.

However, the problem considered in this paper has a special case with both k and $|\Sigma|$ fixed which is not NP-hard unless P=NP. A reasonable direction of future research is to determine if this case is solvable in polynomial time.

References

1. Galant, J., Maier, D., Astorer, J.: On finding minimal length superstrings. Journal of Computer and System Sciences 20(1), 50–58 (1980)
2. Kapun, E., Tsarev, F.: De Bruijn superwalk with multiplicities problem is NP-hard. BMC Bioinformatics 14(suppl. 5), S7 (2013)
3. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of Computer Computation. The IBM Research Symposia Series, pp. 85–103. Plenum Press (1972)
4. Mahaney, S.R.: Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. Journal of Computer and System Sciences 25(2), 130–143 (1982)

5. Medvedev, P., Brudno, M.: Maximum likelihood genome assembly. Journal of Computational Biology 16(8), 1101–1116 (2009)
6. Medvedev, P., Georgiou, K., Myers, G., Brudno, M.: Computability of models for sequence assembly. In: Giancarlo, R., Hannenhalli, S. (eds.) WABI 2007. LNCS (LNBI), vol. 4645, pp. 289–301. Springer, Heidelberg (2007)
7. Medvedev, P., Pham, S., Chaisson, M., Tesler, G., Pevzner, P.: Paired de Bruijn graphs: A novel approach for incorporating mate pair information into genome assemblers. Journal of Computational Biology 18(11), 1625–1634 (2011)
8. Pevzner, P.A., Tang, H., Waterman, M.S.: An eulerian path approach to DNA fragment assembly. Proceedings of the National Academy of Sciences 98(17), 9748–9753 (2001)
9. Pham, S.: Mate-pair consistency and generating problems. In: Talk at RECOMB Satellite Conference on Open Problems in Algorithmic Biology (2012)