On the Complexity of Positional Sequencing by Hybridization

A. Ben-Dor[1], I. Pe'er[2], R. Shamir[2], and R. Sharan[2]

[1] Department of Computer Science and Engineering,
University of Washington, Washington, USA.
amirbd@cs.washington.edu.
[2] Department of Computer Science,
Tel-Aviv University, Tel-Aviv, Israel.
{izik,shamir,roded}@math.tau.ac.il.

Abstract. In sequencing by hybridization (SBH), one has to reconstruct a sequence from its k-long substrings. SBH was proposed as a promising alternative to gel-based DNA sequencing approaches, but in its original form the method is not competitive. Positional SBH is a recently proposed enhancement of SBH in which one has additional information about the possible positions of each substring along the target sequence. We give a linear time algorithm for solving the positional SBH problem when each substring has at most two possible positions. On the other hand, we prove that the problem is NP-complete if each substring has at most three possible positions.

1 Introduction

Sequencing by hybridization (SBH) was proposed and patented in the late eighties as an alternative approach to gel-based sequencing [4, 8, 15, 2, 9]. Using DNA chips, cf. [16], one can in principle determine exactly which k-mers (k-tuples) appear as substrings in a target that needs sequencing, and try to infer its sequence. Practical values of k are 8 to 10.

The fundamental computational problem in SBH is the reconstruction of a sequence from its *spectrum* - the list of all k-mers that are included in the sequence (along with their multiplicities). A naive approach to the problem is to look for a Hamiltonian path in a directed graph whose vertices correspond to k-mers in the spectrum, and two vertices are connected if the $(k-1)$-suffix of one equals the $(k-1)$-prefix of the other. This is however a computationally hard problem. Pevzner [10] has shown that the reconstruction problem can be reduced to finding an Eulerian path in another directed graph, an easily solvable problem. In that graph, vertices correspond to $(k-1)$-tuples, and for each k-tuple in the spectrum, an edge connects the vertices corresponding to its $(k-1)$-long prefix and suffix.

The main handicap of SBH is ambiguity of the solution. Alternative solutions are manifested as *branches* in the graph (i.e., two or more edges leaving the same vertex), and unless the number of branches is very small, there is no good way to determine the correct sequence. Theoretical analysis and simulations [17, 11] have shown that the average length of a uniquely reconstructible sequence using 8-mer chip is only about two hundred, way below a single read length on a commercial gel-lane machine.

Due to the centrality of the sequencing problem in biotechnology and in the Human Genome Project, and due to its mathematical elegance, SBH continues to draw a lot of attention. Many authors have suggested ways to improve the basic method. Alternative chip designs [4, 7, 12, 13] as well as interactive protocols [14] were suggested. An effective and competitive sequencing solution using SBH has yet to be demonstrated.

Recently, several authors have suggested enhancements of SBH based on adding location information to the spectrum [1, 5, 6]. In *positional sequencing by hybridization* (PSBH), additional information is gathered concerning the position of the k-mers in the target sequence. More precisely, for each k-mer in the spectrum its allowed positions along the target are registered. The reduction to the Eulerian path problem still applies, but for each edge in Pevzner's graph we now have constraints restricting its position in the Eulerian path. Mathematically, this gives rise to the *positional Eulerian path* problem (PEP): Given a directed graph with a list of allowed positions on each edge, decide if there exists an Eulerian path in which each edge appears in one of its allowed positions. Hannenhalli et al. [6] showed that PEP is NP-complete, even if all the lists of allowed positions are intervals of equal length. Note that this leaves open the complexity of PSBH. They also gave a polynomial algorithm for the problem when the length of the intervals is bounded.

In this paper we address the positional sequencing by hybridization problem in the case that the number of allowed positions per k-mer is bounded, and the positions need not be consecutive. We give a linear time algorithm for solving the positional Eulerian path problem, and hence, the PSBH problem, in the case that each edge is allowed at most two positions. On the negative side, we show that the problem of PSBH is NP-complete, even if each k-mer has at most three allowed positions and multiplicity one. We use in our hardness proof a reduction from the positional Eulerian path problem restricted to the case where each edge is allowed at most three positions. The latter problem is shown to be NP-complete as well.

The paper is organized as follows: In Section 2 we define the PSBH and the PEP problems. In Section 3 we describe a linear time algorithm for the PEP problem when each edge has at most two allowed positions. In Section 4 we prove that the PEP problem is NP-complete if each edge has at most three allowed positions. Finally, we show in Section 5 that the PSBH problem is NP-complete when each k-mer is allowed at most three positions. For lack of space, some proofs are omitted.

2 Preliminaries

All graphs in this paper are simple, finite, and directed. Let $D = (V, E)$ be a graph. We denote $m = |E|$ throughout. For a vertex $v \in V$, we define its *in-neighbors* to be the set of all vertices from which there is an edge directed into v. We denote this set by $N_{in}(v) = \{u : (u, v) \in E\}$. We define the *in-degree* of v to be $|N_{in}(v)|$. The *out-neighbors* $N_{out}(v)$ and *out-degree* are similarly defined.

Let $E = \{e_1, \ldots, e_m\}$ and let P be a function mapping each edge of D to a non-empty set of integer labels from $\{1 \ldots m\}$ (its *allowed positions*). We call such a pair

(D, P) a *positional graph*. If for all e, $|P(e)| \leq k$, then (D, P) is called a *k-positional graph*. Let $\pi = \pi(1), \ldots, \pi(m)$ be a permutation of the edges in E. If π defines a path, i.e., for each $1 \leq i < m$, $\pi(i) = (u, v)$ and $\pi(i+1) = (v, w)$, for some $u, v, w \in V$, then we say that π is an *Eulerian path* in D.

An Eulerian path π in D is said to be *compliant* with the positional graph (D, P) if $\pi^{-1}(e) \in P(e)$ for each $e \in E$, that is, each edge in π occupies an allowed position. The k-positional Eulerian path problem is defined as follows:

Problem 1 (k-PEP). **Instance:** A k-positional graph (D, P).
Question: Is there an Eulerian path compliant with (D, P)?

Let $\Sigma = \{A, C, G, T\}$. The *p-spectrum* of a string $X \in \Sigma^*$ is the multi-set of all p-long substrings of X. The problem of sequencing by hybridization is defined as follows:

Problem 2 (SBH). **Instance:** A multi-set S of p-long strings.
Question: Is S the p-spectrum of some string X?

For simplicity, we shall call the input multi-set a spectrum, even if it does not correspond to a sequence. The SBH problem is solvable in polynomial time by a reduction to finding an Eulerian path in Pevzner's graph [11]. More specifically, construct a graph D whose vertices correspond to $(p-1)$-long substrings of strings in S, and edges are directed from $\sigma_1 \ldots \sigma_{p-1}$ to $\sigma_2 \ldots \sigma_p$ for each $\sigma_1 \ldots \sigma_p \in S$. Then every solution to the SBH instance naturally corresponds to an Eulerian path in D.

The *positional SBH* problem is defined as follows:

Problem 3 (PSBH). **Instance:** A multi-set S of p-long strings. For each $s \in S$, a set $P(s) \subseteq \{0, \ldots, |S| - 1\}$.
Question: Is S the p-spectrum of some string X, such that for each $s \in S$ its position along X is in $P(s)$?

If the set of allowed positions for each string is of size at most k, then the corresponding problem is called k-positional SBH, or k-PSBH. k-PSBH is linearly reducible to k-PEP in an obvious manner.

3 A Linear Algorithm for 2-Positional Eulerian Path

In this section we provide a linear time algorithm for solving the 2-positional Eulerian path problem. A key element in our algorithm is reducing the problem to 2-SAT. To this end, the input is preprocessed, discarding unrealizable edge labels (positions).

Let $(D = (V, E), P)$ be the input 2-positional graph. For every $1 \leq t \leq m$ define $\Delta(t)$ to be the set of edges allowed at position t, $\Delta(t) \equiv \{e \in E : t \in P(e)\}$. For every vertex $v \in V$ define $In(v, t)$ as the set of t-labeled edges entering v, $In(v, t) \equiv \{(u, v) : (u, v) \in \Delta(t)\}$. Similarly define $Out(v, t) \equiv \{(v, u) : (v, u) \in \Delta(t)\}$.

The first phase of the algorithm applies the following preprocessing step:

```
while $\exists t$ such that $\Delta(t) = \{e\}$ ($\Delta(t)$ is a singleton), do:
    Suppose $P(e) = \{t, t'\}$.
    Set $\Delta(t') \leftarrow \Delta(t') \setminus \{e\}$.
    Set $P(e) \leftarrow \{t\}$.
```

Lemma 1. *The preprocessing step does not change the set of Eulerian paths compliant with* (D, P).

When implementing this step, we maintain current $P(e)$ for each e, and $\Delta(t)$ for each t. If at any stage we discover that some set $\Delta(t)$ is empty, then we output *False* and halt, since no edge can be labeled t. The preprocessing phase can be implemented in linear time. We omit further details. In the following we denote by (D, P) the positional graph obtained after the preprocessing phase. The notation Δ refers to the resulting instance as well.

Lemma 2. *In* (D, P) *each position is allowed for at most two edges.*

Proof. The preprocessing ensures that if for some position t, $|\Delta(t)| = 1$, then $e \in \Delta(t)$ satisfies $|P(e)| = 1$. Let R be the set of positions t with $|\Delta(t)| = 1$, and let $r = |R|$. Then there are $m - r$ positions t for which $|\Delta(t)| \geq 2$, and $r' \geq r$ edges e with $|P(e)| = 1$. Thus,

$$2(m-r) \leq \sum_{t \notin R} |\Delta(t)| = \sum_{t} |\Delta(t)| - r = \sum_{e} |P(e)| - r = 2m - r' - r \leq 2(m-r) \ .$$

Hence, $r = r'$ and each label $t \notin R$ occurs exactly twice, implying that $|\Delta(t)| \in \{1, 2\}$ for all t.■

We say that vertex v is *fixed to position* t in (D, P) if $In(v, t) = \Delta(t)$ or $Out(v, t+1) = \Delta(t+1)$. That is, any Eulerian path compliant with (D, P) must visit v at position t. Define Boolean variables X_e^t for all $t \in P(e)$ ($\sum_e |P(e)| = 2m-r$ variables in total). Define now the following sets of Boolean clauses:

$$X_e^t \quad \text{for every } e \in E \text{ where } P(e) = \{t\} \ . \tag{1}$$

$$X_{e_1}^t \oplus X_{e_2}^t \quad \text{for every } t \notin R \text{ where } \Delta(t) = \{e_1, e_2\} \ . \tag{2}$$

$$X_e^{t_1} \oplus X_e^{t_2} \quad \text{for every } e \in E \text{ where } P(e) = \{t_1, t_2\} \ . \tag{3}$$

$$X_{(a,b)}^t \Leftrightarrow X_{(b,c)}^{t+1} \quad \text{for every } t \in P((a,b)), (t+1) \in P((b,c)), b \text{ is not fixed to } t \ . \tag{4}$$

$$\overline{X}_{(u,v)}^t \quad \text{for every } t \in P((u,v)), t < m, \text{ s.t. } Out(v, t+1) = \emptyset \ . \tag{5}$$

$$\overline{X}_{(u,v)}^t \quad \text{for every } t \in P((u,v)), t > 1, \text{ s.t. } In(u, t-1) = \emptyset \ . \tag{6}$$

Lemma 3. *There is a positional Eulerian path compliant with* (D, P) *iff the set of clauses (1)-(6) is satisfiable.*

Proof. Suppose that a satisfying truth assignment Φ exists. We shall assign an edge e to position t iff $\Phi(X_e^t)$ =*True*. Clauses (1) and (2) guarantee that exactly one edge is assigned to each position. Clauses (1) and (3) guarantee that each edge is assigned to exactly one position, and that this position is allowed to the edge.

It remains to show that the above assignment of edges to positions yields a path in D. Suppose to the contrary that both $X_{(a,b)}^t$ and $X_{(b',c')}^{t+1}$ are assigned *True*, with $b \neq b'$. Then clauses (5) guarantee the existence of an edge $(b, c) \in \Delta(t+1)$, while clauses (6) guarantee the existence of an edge $(a', b') \in \Delta(t)$. Therefore, b is not fixed to t, and a contradiction follows from clauses (4). Hence, Φ defines an Eulerian path compliant with (D, P).

The converse can be shown in a similar way■

Theorem 1. 2-*PEP is solvable in linear time.*

Proof. The preprocessing phase is linear. By Lemma 2 the number of clauses (1)-(6) is $O(m)$. Each XOR clause in (2)-(3) and each equivalence clause in (4) can be written as two OR clauses. Moreover, one can generate all clauses in linear time. By Lemma 3 the problem is reduced to an instance of 2-SAT which is solvable in linear time [3].■

Corollary 1. 2-*PSBH is solvable in linear time.*

4 3-Positional Eulerian Path is NP-Complete

In this section we prove that the 3-PEP problem is NP-complete by reduction from 3-SAT.

Theorem 2. *The* $\mathbf{3}$-*PEP problem is NP-complete*

Proof. Membership in NP is trivial. We prove NP-hardness by reduction from *3-SAT*. Let F be a 3-CNF formula with N variables $x_1, \ldots, x_N$, and M clauses $C_1, \ldots, C_M$. We assume, w.l.o.g., that each clause contains three distinct variables, and that all $2N$ literals occur in F. Denote $X_i = \{x_i\} \cup \{\overline{x}_i\}$. For a literal $L \in X_i$, let a_L denote the number of its occurrences in F. For $1 \leq j \leq a_L$ define $L(j) \equiv (L, j)$, thus $L(1), \ldots, L(a_L)$ is an enumeration of indices to the occurrences of L in F. For a clause $C = L \vee L' \vee L''$ introducing the j-th (j', j'') occurrence of L (L', L'', respectively), we write $C = L(j) \vee L'(j') \vee L''(j'')$. We shall construct a directed graph $D = (V, E)$ and a map P from E to integer sets of size at most $\mathbf{3}$, such that F is satisfiable iff (D, P) has a compliant Eulerian path.

4.1 Outline of the Construction

We now provide a sketch of the main parts of the construction. For each occurrence of a variable in the formula, a *special vertex* is introduced. Special vertices corresponding to the same literal form a *literal path*. Two literal paths of a variable and its negation are connected in parallel to form a *variable subgraph*. For each clause in the formula, the corresponding special vertices are connected by three edges to form a *clause triangle*. Finally, for each special vertex we introduce a triangle incident on it, called its *bypass triangle* (see figure 1).

The sets of allowed positions are chosen so that they force every compliant Eulerian path to visit the literal paths one by one. A compliant Eulerian path corresponds to a satisfying truth assignment. When a special vertex is visited, either its clause triangle, or its bypass triangle are traversed. Traversing the clause triangle while passing through a certain literal's path corresponds to this literal satisfying the clause. We make sure that for one of x_i and $\overline{x}_i$, no clause triangle is visited while passing through its literal path. Eventually, we enable visiting all unvisited bypass triangles.

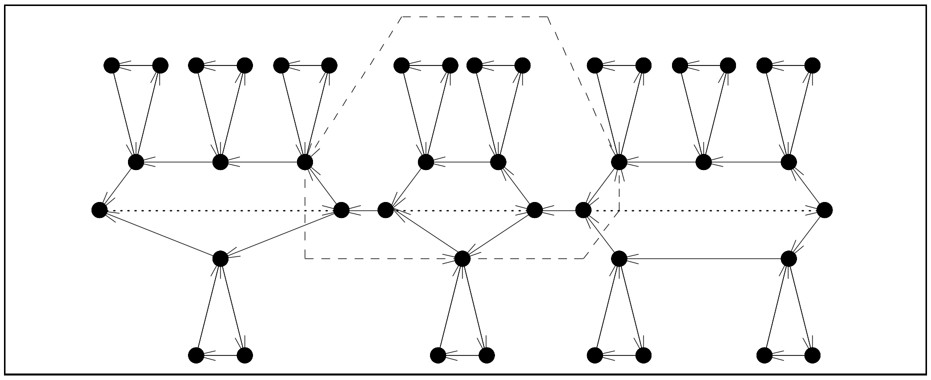

Fig. 1. A schematic sketch of the main elements in our construction. The figure includes three variable subgraphs, with the first variable (whose subgraph is rightmost) having three positive occurrences and two negated occurrences, etc. One of the clause triangles is also drawn, using dashed line.

4.2 Construction in Detail

We introduce the following vertices:

- $u_i, \hat{u}_i$ for each variable x_i, $1 \leq i \leq N$.
- $v_{L(j)}, \hat{v}_{L(j)}$ for each occurrence $L(j)$ of the literal L. $v_{L(j)}$ is called *special*. For $L \in X_i$, we shall denote u_i also by $v_{L(0)}$, and $\hat{u}_i$ also by $v_{L(a_L+1)}$.
- $r(C_c)$ for each clause C_c, $1 \leq c \leq M$, identifying $\hat{u}_N$ as $r(C_0)$ and u_1 as $r(C_{M+1})$.

We introduce the following edges:

- For each clause $C = L(j) \vee L'(j') \vee L''(j'')$, a clause triangle with the edges $\{(v_{L(j)}, v_{L'(j')}), (v_{L'(j')}, v_{L''(j'')}), (v_{L''(j'')}, v_{L(j)})\}$.
- For each occurrence $L(j)$ of the literal L in the clause C, a bypass triangle with the edges $\{(v_{L(j)}, \hat{v}_{L(j)}), (\hat{v}_{L(j)}, r(C)), (r(C), v_{L(j)})\}$.
- A literal path $lpath(L)$: $(u_i, v_{L(1)}), (v_{L(1)}, v_{L(2)}), (v_{L(2)}, v_{L(3)}), \ldots, (v_{L(a_L)}, \hat{u}_i)$, for each literal $L \in X_i$.
- For $i = 1, \ldots, N$, *back* edges $(\hat{u}_i, u_i)$; for $i = 1, \ldots, N-1$, *forward* edges $(\hat{u}_i, u_{i+1})$.
- A *finishing path* $(\hat{u}_N, r(C_1)), (r(C_1), r(C_2)), (r(C_2), r(C_3)), \ldots, (r(C_M), u_1)$.

Figure 2 shows an example of this constructed graph. The motivation for this construction is the following: Using the position sets, we intend to force the literal paths of the different variables to be traversed in the natural order, where the only degree of freedom is switching order between $lpath(x_i)$ and $lpath(\overline{x}_i)$. This switch will correspond to a truth assignment for variable x_i, by assigning *True* to the literal in X_i whose $lpath$ was visited first. After visiting a special vertex along this first path, we either visit its clause triangle, or its bypass triangle. Along the other path (the one of the literal assigned *False*) only a bypass triangle can be visited.

Eventually, the finishing path is traversed. Its vertices are visited in the natural order. Upon visiting a vertex $r(C)$, we visit only one bypass triangle - the yet unvisited triangle among those corresponding to the literals of clause C.

We now describe the sets $P(e)$. We will use the following notation:

$$b_i = a_{x_i} + a_{\overline{x}_i} \quad \text{for } i = 1, \ldots, N \ .$$

$$B_i = \sum_{j=1}^{i} b_j \quad \text{for } i = 0, \ldots, N \ (B_0 = 0) \ .$$

$$Base_L = Base_{\overline{L}} = Base_i = 4B_{i-1} + 4(i-1) \quad \text{for } L \in X_i \ .$$

$$Alternate_L = Base_i + 4a_{\overline{L}} + 2 \quad \text{for } L \in X_i \ .$$

$$ClauseBase_c = Base_{N+1} + 4c \ .$$

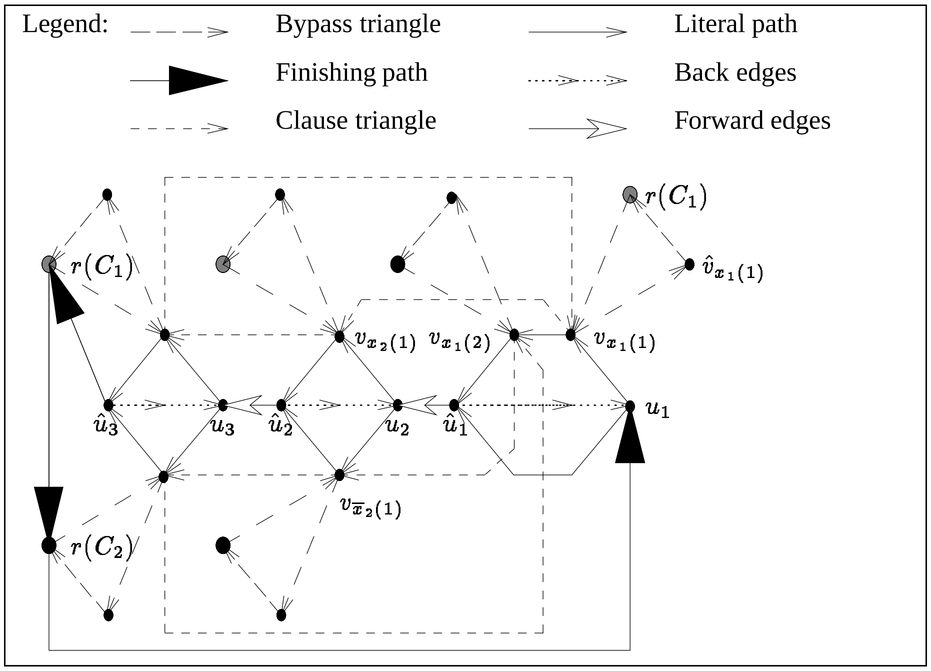

Fig. 2. An example of the construction for the formula $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_3)$. All large grey (black) vertices are actually the same vertex $r(C_1)$ ($r(C_2)$).

– For each forward edge $e = (\hat{u}_{i-1}, u_i)$, $2 \leq i \leq N$, we set $P(e) = \{Base_i\}$. This is intended to ensure that the literal paths are traversed in a constrained order: $lpath(x_i)$ and $lpath(\overline{x}_i)$ are allocated a time interval $[Base_i + 1, Base_{i+1} - 1]$ of length $4b_i + 3$, during which they must be traversed.

- For each back edge $e = (\hat{u}_i, u_i)$ we set $P(e) = \{Alternate_{x_i}, Alternate_{\overline{x}_i}\}$. This enables either visiting $lpath(x_i)$ first, then e and $lpath(\overline{x}_i)$, or visiting $lpath(\overline{x}_i)$ first, followed by e and $lpath(x_i)$.
- For each literal path edge $e = (v_{L(j)}, v_{L(j+1)})$, with $L \in X_i$, $0 \leq j \leq a_L$, we set $P(e) = \{Base_i + 4j + 1, Alternate_L + 4j + 1\}$. Consecutive edges in a literal path are thus positioned 4 time units apart (allowing a triangle in-between).
- For each clause $C = L_1(j_1) \vee L_2(j_2) \vee L_3(j_3)$ with the clause triangle $\{e_1 = (v_{L_1(j_1)}, v_{L_2(j_2)}), e_2 = (v_{L_2(j_2)}, v_{L_3(j_3)}), e_3 = (v_{L_3(j_3)}, v_{L_1(j_1)})\}$ such that $L_k \in X_{i_k}$, define $t_k = Base_{i_k} + 4j_k - 2$ and set

$$P(e_1) = \{t_1, t_3 + 1, t_2 + 2\} ,$$
$$P(e_2) = \{t_2, t_1 + 1, t_3 + 2\} ,$$
$$P(e_3) = \{t_3, t_2 + 1, t_1 + 2\} .$$

  This means that the edges of a clause triangle must be visited consecutively during the traversal of $lpath(L_k)$, for some k. Furthermore, note that this may happen only if $lpath(L_k)$ is traversed immediately after time $Base_{L_k}$, that is, only if it precedes $lpath(\overline{L}_k)$ (see figure 3).

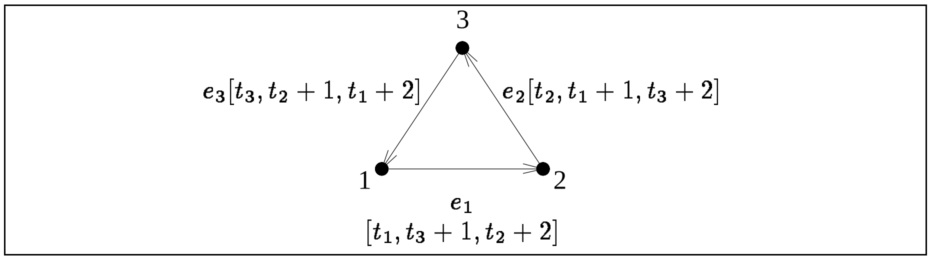
3
$e_3[t_3, t_2 + 1, t_1 + 2]$
$e_2[t_2, t_1 + 1, t_3 + 2]$
1
2
e_1
$[t_1, t_3 + 1, t_2 + 2]$

Fig. 3. A clause triangle, with vertex $v_{L_k(j_k)}$ denoted by k. The allowed positions for each edge appear in brackets.

- For each finishing edge $e = (r(C_c), r(C_{c+1}))$, $0 \leq c \leq M$, we set $P(e) = \{ClauseBase_c\}$, thus determining the order in which the vertices of the finishing path are visited, allowing a time slot $[ClauseBase_c + 1, ClauseBase_{c+1} - 1]$ of length 3, for the bypass triangle visited while traversing $r(C_c)$.
- For a bypass triangle with the edges $\{e = (v_{L(j)}, \hat{v}_{L(j)})$, $e' = (\hat{v}_{L(j)}, r(C_c))$, $e'' = (r(C_c), v_{L(j)})\}$, we set:

$$P(e) = \{Base_L + 4j - 2, Alternate_L + 4j - 2, ClauseBase_{c-1} + 2\} ,$$
$$P(e') = \{Base_L + 4j - 1, Alternate_L + 4j - 1, ClauseBase_{c-1} + 3\} ,$$
$$P(e'') = \{Base_L + 4j, Alternate_L + 4j, ClauseBase_{c-1} + 1\} .$$

  This means that the bypass triangle edges must be visited consecutively, and there are three possible time slots for that:
  - While traversing $lpath(L)$, before traversing $lpath(\overline{L})$.

- While traversing $lpath(L)$, after traversing $lpath(\overline{L})$.
- While traversing $r(C_c)$ along the finishing path.

The reduction is obviously polynomial. We now prove validity of the construction.

$\Leftarrow$ Suppose that F is satisfiable. We will show that (D, P) is a "yes" instance of the 3-positional Eulerian path problem. Let ϕ be a truth assignment satisfying F. For each clause C_c, let $L_c(j_c)$ be a specific literal occurrence satisfying C_c.
We describe an Eulerian path π in D. Set $\pi(ClauseBase_c) = (r(C_c), r(C_{c+1}))$, for $c = 0, \ldots, M$. Set $\pi(Base_i) = (\hat{u}_{i-1}, u_i)$, for $i = 2, \ldots, N$. For all i, if $\phi(x_i)$ =*True*, set $\pi(Alternate_{\overline{x}_i}) = (\hat{u}_i, u_i)$. Otherwise, set $\pi(Alternate_{x_i}) = (\hat{u}_i, u_i)$.
For each literal $L \in X_i$:

- If $\phi(L)$ =*True*: For each $0 \leq j \leq a_L$, set $\pi(Base_i + 4j + 1) = (v_{L(j)}, v_{L(j+1)})$ (see figure 4, top).

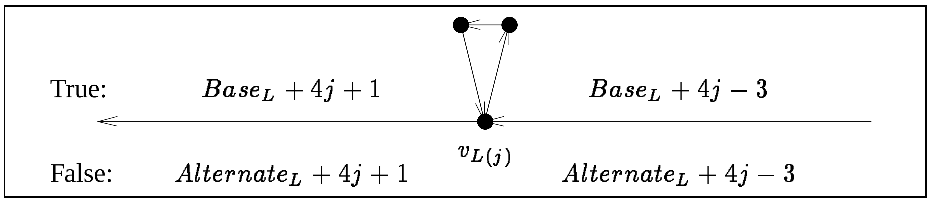

Fig. 4. Either a clause triangle or a bypass triangle must be traversed upon visiting a special vertex $v_{L(j)}$, due to time constraints. Edge positions in case L is assigned *True* (*False*) are shown at the top (bottom).

We further distinguish between two cases:

∗ If $L(j) = L_c(j_c)$ for the clause $C_c = L(j) \vee L'(j') \vee L''(j'')$ in which $L(j)$ occurs, then set π to visit the edges of the clause triangle of C_c as follows:

$$\begin{aligned} \pi(Base_L + 4j - 2) &= (v_{L(j)}, v_{L'(j')}) \ , \\ \pi(Base_L + 4j - 1) &= (v_{L'(j')}, v_{L''(j'')}) \ , \\ \pi(Base_L + 4j) &= (v_{L''(j'')}, v_{L(j)}) \ . \end{aligned}$$

Furthermore, in this case we set π to visit the edges of the bypass triangle of $L(j)$ as follows:

$$\begin{aligned} \pi(ClauseBase_{c-1} + 1) &= (r(C_c), v_{L(j)}) \ , \\ \pi(ClauseBase_{c-1} + 2) &= (v_{L(j)}, \hat{v}_{L(j)}) \ , \\ \pi(ClauseBase_{c-1} + 3) &= (\hat{v}_{L(j)}, r(C_c)) \ . \end{aligned}$$

- $*$ Otherwise, $L(j) \neq L_c(j_c)$ for the clause C_c in which $L(j)$ occurs. In this case we set π to visit the edges of the bypass triangle of $L(j)$ as follows:

$$\begin{aligned}\pi(Base_L + 4j - 2) &= (v_{L(j)}, \hat{v}_{L(j)}) \ , \\ \pi(Base_L + 4j - 1) &= (\hat{v}_{L(j)}, r(C_c)) \ , \\ \pi(Base_L + 4j) &= (r(C_c), v_{L(j)}) \ .\end{aligned}$$

- If $\phi(L)$ =*False*: For each $0 \leq j \leq a_L$, set $\pi(Alternate_L + 4j + 1) = (v_{L(j)}, v_{L(j+1)})$. Furthermore, in this case we set π to visit the edges of the bypass triangle of $L(j)$ as follows (see figure 4, bottom):

$$\begin{aligned}\pi(Alternate_L + 4j - 2) &= (v_{L(j)}, \hat{v}_{L(j)}) \ , \\ \pi(Alternate_L + 4j - 1) &= (\hat{v}_{L(j)}, r(C_c)) \ , \\ \pi(Alternate_L + 4j) &= (r(C_c), v_{L(j)}) \ .\end{aligned}$$

Examining all the cases shows that π is a permutation of the edges, and if $\pi(k) = (u, v)$,$\pi(k+1) = (u', v')$ then $v = u'$. Hence, π is an Eulerian path. Furthermore, by our construction π is compliant with (D, P), proving that (D, P) is a ”yes” instance.

$\Rightarrow$ Let π be an Eulerian path compliant with (D, P). We shall construct an assignment ϕ satisfying F. In order to determine $\phi(x_i)$ we consider the edge $\pi(Base_i + 1)$. By construction, $\pi(Base_i + 1) = (u_i, v_{L(1)})$ for $L \in X_i$. We therefore set $\phi(L)$ =*True* (and of course $\phi(\overline{L})$ =*False*). We observe that for any other edge $e' = (v_{L(j)}, v_{L(j+1)})$ along $lpath(L)$, we must have $\pi(Base_i + 4j + 1) = e'$ iff $\phi(L)$ =*True*.
We now prove that ϕ satisfies each clause $C_c = L_1(j_1) \vee L_2(j_2) \vee L_3(j_3)$ in F. Consider the clause triangle of C_c: $\{e_1 = (v_{L_1(j_1)}, v_{L_2(j_2)})$, $e_2 = (v_{L_2(j_2)}, v_{L_3(j_3)})$, $e_3 = (v_{L_3(j_3)}, v_{L_1(j_1)})\}$. Denote $t_k = Base_{L_k} + 4j_k - 2$. Suppose that $\pi^{-1}(e_k) \neq t_k$, for some $1 \leq k \leq 3$, then by the positional constraints the edge visited prior to e_k must be in the clause triangle. Therefore, there exists some $1 \leq k \leq 3$ for which $\pi(t_k) = e_k$. Furthermore, the edge e preceding e_k in π must have $t_k - 1 = Base_{L_k} + 4(j_k - 1) + 1 \in P(e)$. The only such edge entering $v_{L_k(j_k)}$ is the literal path edge $(v_{L_k(j_k-1)}, v_{L_k(j_k)})$. Therefore, $\phi(L_k)$ =*True*, satisfying C_c.

This proves that F is satisfiable iff (D, P) is a ”yes” instance, completing the proof of Theorem 2.∎

Observe that the graph constructed in the proof of Theorem 2 has in-degree and out-degree bounded by 4, giving rise to the following result:

Corollary 2. *3-PEP is NP-complete, even when restricted to graphs with in-degree and out-degree bounded by 4 .*

Henceforth, we call this restricted problem *(3,4)-PEP*. We comment that a slight modification of the construction results in a graph whose in-degree and out-degree are bounded by 2.

5 3-Positional SBH is NP-Complete

We show in this section that the problem of sequencing by hybridization with at most 3 positions per spectrum element is NP-complete, even if each element in the spectrum is unique. The proof is by reduction from (3,4)-PEP.

Theorem 3. *The 3-PSBH problem is NP-complete, even if all spectrum elements are of multiplicity one.*

Proof. It is easy to see that the problem is in NP. We reduce (3,4)-PEP to 3-PSBH. Let $(D = (V, E), P)$ be an instance of (3,4)-PEP. Let $k = \lceil \log_4 |V| \rceil + 2$, $p = 3k + 1$ and $c = p + 1$. In order to construct an instance of 3-PSBH we first encode the edges and vertices of D. In the following, we denote string concatenation by $|$. We let σ_1 ='A', σ_2 ='C', σ_3 ='G' and σ_4 ='T'.

For each $v \in V$ we assign a unique string in Σ^{k-2}. We add a leading 'T' symbol and a trailing 'T' symbol to this string, and call the resulting k-long sequence the *name* of v. We also assign the (unique) sequence 'A...A' of length k to encode a *space*. Each vertex is encoded by a $3k$-long sequence containing two copies of its name separated by a space. We denote the encoding of v by $en(v)$. Each edge $(u, v) \in E$ is encoded by two symbols chosen as follows: Let $N_{out}(u) = \{v_1, \ldots, v_l\}$, where $v = v_i$ for some i, and $l \leq 4$. Let $N_{in}(v) = \{u_1, \ldots, u_r\}$, where $u = u_j$ for some j, and $r \leq 4$. Then (u, v) is encoded by $\sigma_i | \sigma_j$, and we denote its encoding by $en(u, v)$. We let $EN((u, v)) = en(u)|en(u, v)|en(v)$ (see figure 5).

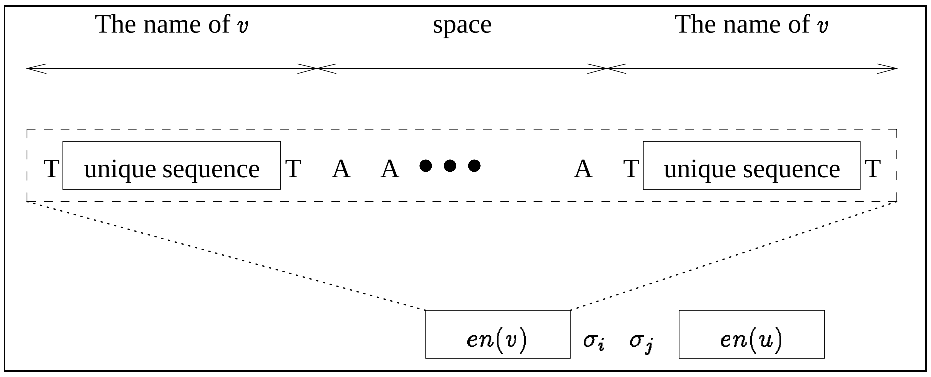

Fig. 5. The encoding of vertices and edges into strings.

We now construct a 3-PSBH instance, i.e., a set S with position constraints T, as follows: For every edge $(u, v) \in E$ the set S contains all p-long substrings of the $2p$-long sequence $EN((u, v))$ (c substrings in total). Let $s^i_{(u,v)}$ denote the i'th substring, $i = 0, \ldots, p$. Let $P((u, v)) = \{t_1, \ldots, t_l\}$, $1 \leq l \leq 3$, be the set of allowed positions for (u, v). Then we set $T(s^i_{(u,v)}) = \{c(t_1 - 1) + i, \ldots, c(t_l - 1) + i\}$ for all i (substring positions are 0-up).

Lemma 4. *Each of the p-long substrings defined above is unique.*

Proof. Suppose that $s = s^i_{(u,v)} = s^j_{(w,z)}$. We first claim that $i = j$. There are two cases to examine: If s contains a space at positions $r, \dots, r + k - 1$, $0 \leq r \leq 2k + 1$, then $i = j = (c + k - r) \pmod{c}$. Otherwise, s begins with a run of 'A' symbols of length $0 \leq r' \leq k - 1$. This run belongs to a space in $en(u)$ and $en(w)$, and must be followed by the symbol 'T'. In this case $i = j = 2k - r'$.

By construction, s contains a name of a vertex plus a unique symbol identifying an edge entering or leaving that vertex, implying that $(u, v) = (w, z)$.■

We now show the validity of the reduction.

- $\Leftarrow$ Suppose that $\pi = (v_0, v_1), (v_1, v_2), \dots, (v_{m-1}, v_m)$ is a solution of the (3,4)-PEP instance. We claim that $X = en(v_0)|en((v_0, v_1))|en(v_1)|en((v_1, v_2))|\dots|en(v_m)$ is a solution of the 3-PSBH instance. By Lemma 4 each p-long substring of X occurs exactly once in X. As π visits all edges in D, we have that S is the p-spectrum of X. The fact that position constraints are obeyed follows directly from the construction.
- $\Rightarrow$ Let X be a solution of the 3-PSBH instance. Consider the m substrings of length p, whose starting positions are integer multiples of c. By the position constraints, the r-th such substring is an encoding of some vertex v_r, followed by a symbol σ_{i_r}. Denote by w_r the i_r-th out-neighbor of v_r. We prove that $\pi = (v_1, w_1), \dots, (v_m, w_m)$ is an Eulerian path compliant with (D, P).
  Since each string in the p-spectrum of X is unique, π is a permutation of the edges in D. To prove that π is a path in D we have to show that $w_r = v_{r+1}$ for $r = 1, \dots, m - 1$. Let x be the p-long substring of X starting at position $rc + 2k$. We observe that x must begin with the last k symbols of $en(v_r)$, which compose $name(v_r)$, followed by σ_{i_r}, some symbol, and the first $2k-1$ symbols of $en(v_{r+1})$, which contain $name(v_{r+1})$. The uniqueness of $name(v_r)$, $name(v_{r+1})$ and the index i_r among the out-neighbors of v_r, implies that $w_r = v_{r+1}$. The claim now follows, since position constraints are trivially satisfied by π.■

Acknowledgments

The first author was supported by the program for mathematics and molecular biology. The second author was supported by the Clore foundation scholarship. The third author was supported in part by a grant from the Ministry of Science, Israel. The fourth author was supported by Eshkol scholarship from the Ministry of Science, Israel.

References

1. L. M. Adleman. Location sensitive sequencing of DNA. Technical report, University of Southern California, 1998.
2. R. Drmanac amd R. Crkvenjakov, 1987. Yugoslav Patent Application 570.
3. B. Apsvall, M. F. Plass, and R.E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

4. W. Bains and G. C. Smith. A novel method for nucleic acid sequence determination. *J. Theor. Biology*, 135:303–307, 1988.
5. S. D. Broude, T. Sano, C. S. Smith, and C. R. Cantor. Enhanced DNA sequencing by hybridization. *Proc. Nat. Acad. Sci. USA*, 91:3072–3076, 1994.
6. S. Hannenhalli, P. Pevzner, H. Lewis, and S. Skiena. Positional sequencing by hybridization. *Computer Applications in the Biosciences*, 12:19–24, 1996.
7. K. R. Khrapko, Yu. P. Lysov, A. A. Khorlyn, V. V. Shick, V. L. Florentiev, and A. D. Mirzabekov. An oligonucleotide hybridization approach to DNA sequencing. *FEBS letters*, 256:118–122, 1989.
8. Y. Lysov, V. Floretiev, A. Khorlyn, K. Khrapko, V. Shick, and A. Mirzabekov. DNA sequencing by hybridization with oligonucleotides. *Dokl. Acad. Sci. USSR*, 303:1508–1511, 1988.
9. S. C. Macevics, 1989. International Patent Application PS US89 04741.
10. P. A. Pevzner. l-tuple DNA sequencing: computer analysis. *J. Biomol. Struct. Dyn.*, 7:63–73, 1989.
11. P. A. Pevzner and R. J. Lipshutz. Towards DNA sequencing chips. In *Symposium on Mathematical Foundations of Computer Science*, pages 143–158. Springer, 1994. LNCS vol. 841.
12. P. A. Pevzner, Yu. P. Lysov, K. R. Khrapko, A. V. Belyavsky, V. L. Florentiev, and A. D. Mirzabekov. Improved chips for sequencing by hybridization. *J. Biomol. Struct. Dyn.*, 9:399–410, 1991.
13. F. Preparata, A. Frieze, and Upfal E. On the power of universal bases in sequencing by hybridization. In *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB '99)*, pages 295–301, 1999.
14. S. S. Skiena and G. Sundaram. Reconstructing strings from substrings. *J. Comput. Biol.*, 2:333–353, 1995.
15. E. Southern, 1988. UK Patent Application GB8810400.
16. E. M. Southern. DNA chips: analysing sequence by hybridization to oligonucleotides on a large scale. *Trends in Genetics*, 12:110–115, 1996.
17. E. M. Southern, U. Maskos, and J. K. Elder. Analyzing and comparing nucleic acid sequences by hybridization to arrays of oligonucleotides: evaluation using experimental models. *Genomics*, 13:1008–1017, 1992.