JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 8, Number 4, 2001
Mary Ann Liebert, Inc.
Pp. 361–371

# On the Complexity of Positional Sequencing by Hybridization

A. BEN-DOR,[1] I. PE'ER,[2] R. SHAMIR,[2] and R. SHARAN[2]

## ABSTRACT

**In sequencing by hybridization (SBH), one has to reconstruct a sequence from its *l*-long substrings. SBH was proposed as an alternative to gel-based DNA sequencing approaches, but in its original form the method is not competitive. Positional SBH (PSBH) 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 PSBH 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. We also show that PSBH is NP-complete if the set of allowed positions for each substring is an interval of length *k* and provide a fast algorithm for the latter problem when *k* is bounded.**

**Key words:** Positional sequencing by hybridization, complexity, Eulerian graphs, NP-hardness, parameterized algorithms.

## 1. INTRODUCTION

Sequencing by hybridization (SBH) was proposed in the late eighties as an alternative to gel-based DNA sequencing (Bains and Smith, 1988; Lysov *et al.*, 1988; Southern, 1988; Drmanac and Crkvenjakov, 1987; Macevics, 1989). Using DNA chips, cf. Southern (1996), one can in principle determine exactly which $l$-mers ($l$-tuples) appear as substrings in a target that needs sequencing and try to infer its sequence. Practical values of $l$ are 8 to 10.

The fundamental computational problem in SBH is the reconstruction of a sequence from its *spectrum*—the list of all $l$-mers that are included in the sequence along with their multiplicities. It was shown by Pevzner (1989) that the reconstruction problem can be solved efficiently by a reduction to finding an Eulerian path in the following graph: Vertices correspond to $(l-1)$-tuples, and for each $l$-tuple in the spectrum, an edge is directed from the vertex corresponding to its $(l-1)$-long prefix to that of its $(l-1)$-long 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 (Southern *et al.*, 1992; Pevzner and Lipshutz, 1994; Arratia *et al.*, 1997; Dyer *et al.*, 1994) have shown that

[1]Agilent Laboratories and University of Washington.
[2]School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel.

the average length of a uniquely reconstructible sequence using an 8-mer chip is only about two hundred, far 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 (Bains and Smith, 1988; Khrapko *et al.*, 1989; Pevzner *et al.*, 1991; Preparata *et al.*, 1999), usage of prior sequence information (Pe'er and Shamir, 2000), as well as interactive protocols (Skiena and Sundaram, 1995; Frieze and Halldorsson, 2001) 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 (Adleman, 1998; Broude *et al.*, 1994; Hannenhalli *et al.*, 1996; Gusfield *et al.*, 1998; Shamir and Tsur, 2001). In *positional sequencing by hybridization* (PSBH), additional information is gathered concerning the position of the $l$-mers in the target sequence. More precisely, for each $l$-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 for each edge, decide if there exists an Eulerian path in which each edge appears in one of its allowed positions. Observe that the graph theoretic problem, PEP, is more general than PSBH, since it assumes no string assignment to vertices and edges. Hannenhalli *et al.* (1996) 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 the more restricted PSBH in this case. They also gave a polynomial algorithm for the problem when the length of the intervals is bounded.

In this paper, we first address the positional sequencing by hybridization problem in the case that the number of allowed positions per $l$-mer is bounded and the positions need not be consecutive. The case of nonconsecutive positions is mainly of theoretical interest, though some application has been suggested by Adleman (1998). After reviewing definitions (Section 2), 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 (Section 3). On the negative side, we prove the NP-completeness of PEP, even when restricted to the case where each edge is allowed at most three positions (Section 4). We conclude that PSBH is NP-complete, even if each $l$-mer has at most three allowed positions and multiplicity one (Section 5). We also study the complexity of PSBH in the case that the set of allowed positions for each substring is an interval of bounded length. We strengthen the results of Hannenhalli *et al.* (1996) with respect to this problem in two ways. First, we show that PSBH (and not only PEP) is NP-complete, even if all sets of allowed positions are $k$-long intervals (Section 5). Second, we give a faster parameterized algorithm for the problem, where the parameter $k$ is an upper bound on the size of the intervals (Section 6). Our algorithm requires $O(mk^{1.5}4^k)$ time, compared to the $O(mk^3 \log k 4^k)$ bound of Hannenhalli *et al.* (1996). For brevity, some proof details are omitted. Full details are given in Ben-Dor *et al.* (2001).

## 2. PRELIMINARIES

All graphs in this paper are 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 nonempty set of integer labels from $\{1, \ldots, m\}$. The set $P(e)$ is called the set of *allowed positions* of edge $e$. The pair $(D, P)$ is called 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 $\pi$ defines a (directed) 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 $\pi$ is an *Eulerian path* in $D$.

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

**Problem 1 (PEP)**
**Instance:** *A positional graph* $(D, P)$.
**Question:** *Is there an Eulerian path which complies with* $(D, P)$*?*

If each $P(e)$ is a subinterval of $[1, m]$, then the problem is called *Interval PEP*. If $(D, P)$ is a $k$-positional graph then the problem is called *k-positional Eulerian path* ($k$-PEP).

Let $\sum = \{A, C, G, T\}$. The *p-spectrum* of a string $X \in \sum^*$ 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 (Pevzner and Lipshutz, 1994). Specifically, construct a graph $D$ whose vertices correspond to $(p-1)$-long substrings of strings in $S$ and in which edges are directed from $\sigma_1 \cdots \sigma_{p-1}$ to $\sigma_2 \cdots \sigma_p$ for each $\sigma_1 \cdots \sigma_p \in S$. Then every SBH solution $\sigma_1 \cdots \sigma_{m+p-1}$ naturally corresponds to an Eulerian path $\sigma_1 \cdots \sigma_{p-1}, \cdots, \sigma_{m+1} \cdots \sigma_{m+p-1}$ 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*, which is linearly reducible to $k$-PEP in an obvious manner. If, for each $s \in S$, $P(s)$ is a sub-interval of $[0, |S| - 1]$, the problem is called *Interval PSBH*.

## 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 a reduction to 2-SAT. Before the reduction can be applied, the input must be 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 $t$, i.e., $\Delta(t) \equiv \{e \in E : t \in P(e)\}$. We call a position $t$ for which $\Delta(t) = \{e\}$ and $|P(e)| > 1$, a *resolvable* position.

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

> **While** there exists a resolvable position $t$, **do**:
> Suppose $\Delta(t) = \{e\}$ and $P(e) = \{t, t'\}$.
> $\Delta(t') \leftarrow \Delta(t') \backslash \{e\}$.
> $P(e) \leftarrow \{t\}$.

If at any stage we find that some set $\Delta(t)$ is empty, we output *False* and halt, as no edge can be labeled $t$.

**Lemma 1.** *The preprocessing step does not change the set of Eulerian paths which comply with* $(D, P)$*. This step can be implemented in linear time.*

In the following, $(D, P)$ and $\Delta$ refer to the positional graph obtained after the preprocessing phase.

**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 $t$. ■

For every vertex $v \in V$, define $In(v, t)$ as the set of $t$-labeled edges entering $v$, i.e., $In(v, t) \equiv \{(u, v) : (u, v) \in \Delta(t)\}$. Similarly, define $Out(v, t) \equiv \{(v, u) : (v, u) \in \Delta(t)\}$. We say that a vertex $v$ is *fixed to position $t$* in $(D, P)$ if $In(v, t) = \Delta(t)$ or $Out(v, t + 1) = \Delta(t + 1)$. In other words, any Eulerian path compliant with $(D, P)$ must have $v$ as the $(t + 1)$-st vertex in the path. Define Boolean variables $X_e^t$ for every $e$, $t$ such that $t \in P(e)$ ($2m - r$ variables in total). Examine the following sets of Boolean clauses:

$$X_e^t \quad \text{for every } e, t \text{ such that } P(e) = \{t\}. \tag{1}$$

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

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

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

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

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

**Lemma 3.** *There is a positional Eulerian path which complies with $(D, P)$ if and only if the set of clauses (1)–(6) is satisfiable.*

**Proof.** The proof is trivial, since a positional Eulerian path induces an obvious truth assignment and vice versa. ■

**Theorem 4.** *2-PEP is solvable in linear time.*

**Proof.** The preprocessing phase is linear by Lemma 1. Clearly, the number of clauses (1)–(6) is $O(m)$. Each exclusive OR clause in (2)–(3) and each equivalence clause in (4) can be written as two OR clauses. It can be shown that generating all clauses takes linear time. By Lemma 3, the problem is reduced to an instance of 2-SAT which is solvable in linear time (Apsvall *et al.*, 1979). ■

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

## 4. 3-POSITIONAL EULERIAN PATH IS NP-COMPLETE

In this section we prove that 3-PEP is NP-complete.

**Theorem 5.** *3-PEP is NP-complete.*

**Proof.** Membership in NP is trivial. We prove NP-hardness by reduction from *3-SAT*. We first provide a sketch of the construction. For each occurrence of a literal in the formula, a *special vertex* is introduced. Special vertices corresponding to the same literal are connected serially to 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 Fig. 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 is traversed. Traversing the clause triangle while passing through a certain literal's path corresponds to this literal satisfying the clause. Eventually, we enable visiting all unvisited bypass triangles.

We now give the construction in detail. 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 $1 \le j \le a_L$, define $L(j) \equiv (L, j)$. Thus, $L(1), \ldots, L(a_L)$ is an enumeration of $L$'s occurrences 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 three, such that $F$ is satisfiable if and only if $(D, P)$ has a compliant Eulerian path. We introduce the following vertices:

- $u_i$, $\hat{u}_i$ for each variable $x_i$, $1 \le i \le N$.
- $v_{L(j)}$, $\hat{v}_{L(j)}$ for each occurrence $L(j)$ of the literal $L$. We call $v_{L(j)}$ *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 $C_c$, $1 \le c \le M$, identifying $\hat{u}_N$ as $r(C_0)$ and $u_1$ as $r(C_{M+1})$.

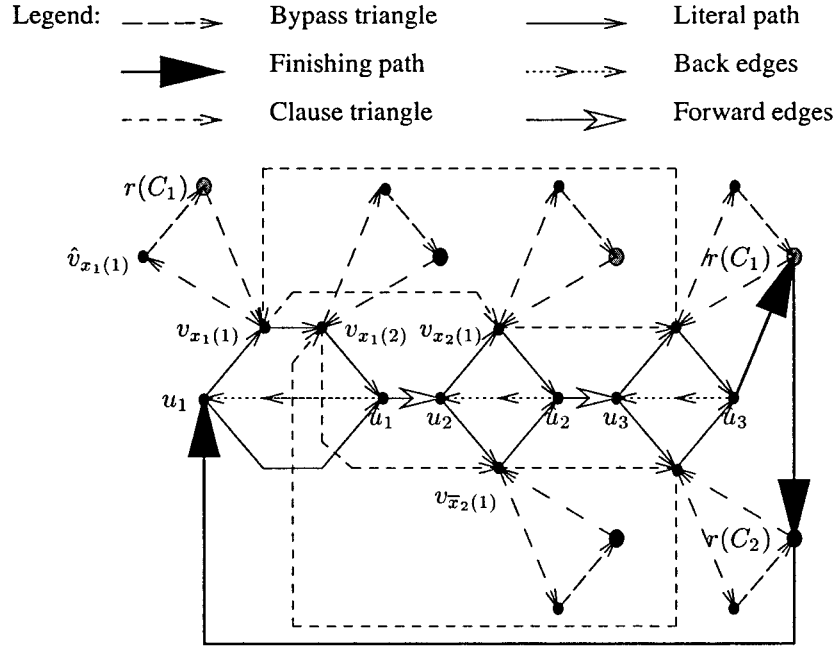


**FIG. 1.** An example of the construction for the formula $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_3)$. The variable subgraphs are at the center; bypass triangles are at the top and bottom parts. Different bypass triangles of the same clause use the same clause vertex, but for clarity this vertex is drawn multiple times. For example, the two large vertices denoted $r(C_1)$ and the large vertex in the bypass triangle of $v_{x_2}(1)$ are actually the same vertex. The figure includes three variable subgraphs. The first variable, $x_1$, whose subgraph is the leftmost, has two positive occurrences. Each of the two other variables has a single positive occurrence and a single negated one.

We also introduce the following edges:

- For each clause $C = L(j) \vee L'(j') \vee L''(j'')$, a clause triangle consisting of 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 a literal $L$ in a 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 $lp(L)$: $\{(u_i, v_{L(1)}), (v_{L(1)}, v_{L(2)}), (v_{L(2)}, v_{L(3)}), \ldots, (v_{L(a_L)}, \hat{u}_i)\}$ 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 1 shows an example of the 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 the order between $lp(x_i)$ and $lp(\overline{x}_i)$. This switch will correspond to a truth assignment for the variable $x_i$, by assigning *True* to the literal in $X_i$ whose path was visited first. After visiting a special vertex along this first path, we visit either its clause triangle or its bypass triangle. Along the other path (of the literal assigned *False*), only a bypass triangle can be visited.

Eventually, the finishing path is traversed. Upon visiting a vertex $r(C)$, we visit only one bypass triangle—the yet unvisited triangle among those corresponding to the literals of clause $C$. The truth assignment will satisfy that literal.

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

$$Base_L = Base_{\overline{L}} = Base_i = 4\sum_{j=1}^{i}(a_{x_i} + a_{\overline{x}_i}) + 4(i-1) \qquad \text{for } L \in X_i.$$

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

$$ClauseBase_c = Base_{N+1} + 4c \qquad 0 \le c \le M.$$

- For each forward edge $e = (\hat{u}_{i-1}, u_i)$, $2 \le i \le N$, we set $P(e) = \{Base_i\}$. This is intended to ensure that the literal paths are traversed in a constrained order: $lp(x_i)$ and $lp(\overline{x}_i)$ are allocated a time interval $[Base_i + 1, Base_{i+1} - 1]$ of length $4(a_{x_i} + a_{\overline{x}_i}) + 3$ during which they must be traversed.
- For each back edge $e = (\hat{u}_i, u_i)$, we set $P(e) = \{Alternate_{\overline{x}_i}, Alternate_{x_i}\}$. This enables either visiting $lp(x_i)$ first, then $e$ and $lp(\overline{x}_i)$; or visiting $lp(\overline{x}_i)$ first, followed by $e$ and $lp(x_i)$.
- For each literal path edge $e = (v_{L(j)}, v_{L(j+1)})$, with $L \in X_i$, $0 \le j \le a_L$, we set $P(e) = \{Base_i + 4j + 1, Alternate_L + 4j + 1\}$. Consecutive edges in a literal path are thus positioned four time units apart.
- 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 $t_k \equiv Base_{i_k} + 4j_k - 2$ 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 $lp(L_k)$ for some $k$. Furthermore, note that this may happen only if $lp(L_k)$ is traversed immediately after time $Base_{L_k}$, that is, only if it precedes $lp(\overline{L}_k)$.
- For each finishing edge $e = (r(C_c), r(C_{c+1}))$, $0 \le c \le M$, we set $P(e) = \{ClauseBase_c\}$, thus determining the order of visiting the vertices of the finishing path, allowing a time slot $[ClauseBase_c + 1, ClauseBase_{c+1} - 1]$ of length three for the bypass triangle visited while traversing $r(C_c)$.
- For a bypass triangle with edges $\{e = (v_{L(j)}, \hat{v}_{L(j)}), e' = (\hat{v}_{L(j)}, r(C_c)), e'' = (r(C_c), v_{L(j)})\}$, set:

$$P(e) = \{Base_L + 4j - 2, Alternate_L + 4j - 2, ClauseBase_c - 2\},$$

$$P(e') = \{Base_L + 4j - 1, Alternate_L + 4j - 1, ClauseBase_c - 1\},$$

$$P(e'') = \{Base_L + 4j, Alternate_L + 4j, ClauseBase_c - 3\}.$$

This implies that the bypass triangle edges must be visited consecutively, and there are three possible time slots for that: 1) While traversing $lp(L)$, before traversing $lp(\overline{L})$; 2) while traversing $lp(L)$, after traversing $lp(\overline{L})$; 3) 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 $\phi$ 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 the Eulerian path $\pi$ 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 $\pi(Alternate_{x_i}) = (\hat{u}_i, u_i)$. Consider a 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 Fig. 2, top). 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 $\pi$ at times $Base_L + 4j - 2, \ldots, Base_L + 4j$ to visit the edges of the clause triangle of $C_c$ starting (and ending) at $v_{L(j)}$. Furthermore, in this case, we set $\pi$ at times $ClauseBase_c - 3, \ldots, ClauseBase_c - 1$ to visit the edges of the bypass triangle of $L(j)$ starting at $r(C_c)$.
* Otherwise, $L(j) \neq L_c(j_c)$ for the clause $C_c$ in which $L(j)$ occurs. In this case, we set $\pi$ at times $Base_L + 4j - 2, \ldots Base, L + 4j$ to visit the edges of the bypass triangle of $L(j)$ starting at $v_{L(j)}$.

—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 $\pi$ at times $Alternate_L + 4j - 2, \ldots, Alternate_L + 4j$ to visit the edges of the bypass triangle of $L(j)$ starting at $v_{L(j)}$ (see Fig. 2, bottom).

It is easy to see that $\pi$ is an Eulerian path. Furthermore, by our construction, $\pi$ complies with $(D, P)$, proving that $(D, P)$ is a "yes" instance.

$\Rightarrow$ Let $\pi$ be an Eulerian path compliant with $(D, P)$. We shall construct an assignment $\phi$ 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 $\phi(\overline{L}) = False$). We observe that, for any other edge $e' = (V_{L(j)}, v_{L(j+1)})$ along $lp(L)$, we must have $\pi(Base_i + 4j + 1) = e'$ if and only if $\phi(L) = True$.

We now prove that $\phi$ satisfies each clause $C_c = L_1(j_1) \vee L_2(j_2) \vee L_3(j_3)$ of $F$. 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 $t_k = Base_{L_k} + 4j_k - 2$. By the positional constraints, there exists some $1 \leq k \leq 3$ for which $\pi(t_k) = e_k$. The edge $e$ preceding $e_k$ in $\pi$ 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 $(v_{L_k(j_k-1)}, v_{L_k(j_k)})$. Therefore, $\phi(L_k) = True$, satisfying $C_c$.

Hence, $F$ is satisfiable if and only if $(D, P)$ is a "yes" instance, completing the proof of Theorem 5. ■

**Corollary 2.** *3-PEP is NP-complete, even on graphs with all in-degrees and out-degrees at most four.*

Henceforth, we call this restricted problem *(3,4)-PEP*. We comment that a slight modification of the construction results in a graph whose in-degrees and out-degrees are at most two.

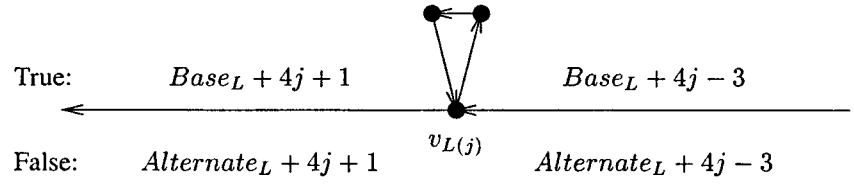


**FIG. 2.** Either a clause triangle or a bypass triangle must be traversed upon visiting a special vertex $v_{L(j)}$, due to time constraints. Edge positions in case $L$ assigned *True* (*False*) are shown at the top (bottom).

## 5. HARDNESS OF POSITIONAL SBH

We show in this section that PSBH with at most 3 positions per spectrum element is NP-complete, even if each element in the spectrum is unique. We also prove the NP-completeness of Interval PSBH, settling a problem that was left open by Hannenhalli *et al.* (1996).

**Theorem 6.** *3-PSBH is NP-complete, even if all spectrum elements are of multiplicity one.*

**Proof.** Clearly, 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 $\sigma_1 =$ "A," $\sigma_2 =$ "C," $\sigma_3 =$ "G," and $\sigma_4 =$ "T."

To each $v \in V$, we assign a distinct string in $\sum^{k-2}$. We add a leading "T" symbol and a trailing "T" symbol to this string and call the resulting $k$-long string the *name* of $v$. We also assign the string "A ... A" of length $k$ to encode a *space*. Each vertex is encoded by a $3k$-long string 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 call $EN(u, v) \equiv en(u)|en(u, v)|en(v)$ the *representative string* of $(u, v)$ (see Fig. 3).

We now construct a 3-PSBH instance, i.e., a spectrum $S$ 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 string $EN(u, v)$ ($c$ substrings in total). Let $s^i_{(u,v)}$ denote the $i$-th such 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 $i$ (note that substring positions are numbered starting at zero).

We now show validity of the reduction.

$\Leftarrow$ Suppose that $\pi = (v_0, v_1), (v_1, v_2), \ldots, (v_{m-1}, v_m)$ is a solution of the (3,4)-PEP instance. We claim that $X = en(v_0)|en(v_0, v_1)|en(v_1)|en(v_1, v_2)|en(v_2)| \ldots |en(v_{m-1})|en(v_{m-1}, v_m)|en(v_m)$ is a solution of the 3-PSBH instance. It is not hard to show that each of the $p$-long substrings in $S$ is unique and, therefore, each $p$-long substring of $X$ occurs exactly once in $X$. As $\pi$ visits all edges in $D$, we have that $S$ is the $p$-spectrum of $X$. The fact that position constraints are satisfied 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 in $X$ 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 $\sigma_{i_r}$. Denote by $w_r$ the $i_r$-th out-neighbor of $v_r$. We prove that $\pi = (v_1, w_1), \ldots, (v_m, w_m)$ is an Eulerian path compliant with $(D, P)$.

Since each string in the $p$-spectrum of $X$ is unique, $\pi$ is a permutation of the edges in $D$. To prove that $\pi$ is a path in $D$ we have to show that $w_r = v_{r+1}$ for $r = 1, \ldots, m - 1$. Let $x$ be the $p$-long substring

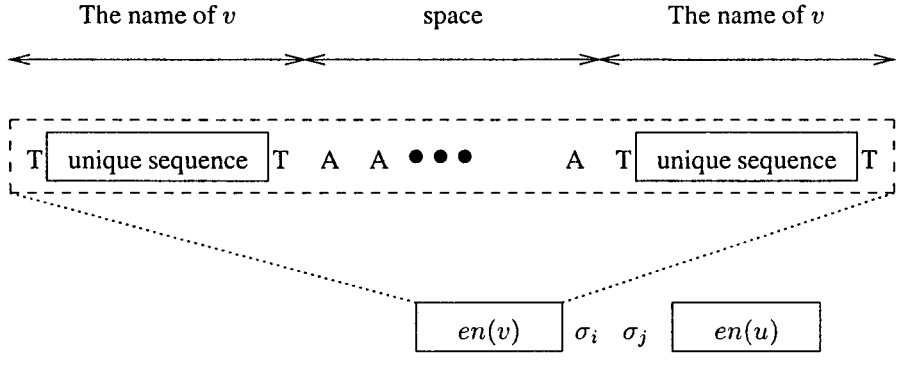


**FIG. 3.** The encoding of vertices and edges into representative strings.

of $X$ starting at position $(r-1)c+2k$. We observe that $x$ must begin with the last $k$ symbols of $en(v_r)$, which compose $name(v_r)$, followed by $\sigma_{i_r}$, 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 $\pi$. ■

**Theorem 7.** *The Interval PSBH problem is NP-complete, even if all sets of allowed positions are intervals of equal length.*

**Proof.** Interval PEP is NP-complete, even if each vertex has in-degree and out-degree at most two, and the sets of allowed positions are intervals of equal length (Hannenhalli *et al.*, 1996). This restriction of Interval PEP is reducible to Interval PSBH. The reduction is analogous to the one used in the proof of Theorem 6. ■

## 6. A PARAMETERIZED ALGORITHM FOR INTERVAL PSBH

Hannenhalli *et al.* have given a linear-time algorithm to solve the parametric version of Interval PEP (and hence, Interval PSBH), where the parameter $k$ is an upper bound on the sizes of the intervals of allowed positions for each edge (Hannenhalli *et al.*, 1996). We provide a faster algorithm for the problem, which uses the same basic idea. Our algorithm runs in $O(mk^{1.5}4^k)$ time, improving upon the $O(mk^3 \log k 4^k)$ time bound of Hannenhalli *et al.* (1996).

For simplicity, we shall describe our algorithm when all allowed intervals have length exactly $k$. Let $(D=(V,E),P)$ be the input positional graph, where $P(e)=[l_e, l_e+k-1]$ for every $e \in E$. For every $1 \le i \le m$, define $\underline{i} \equiv \max\{i-k+1, 1\}$ and $\overline{i} \equiv \min\{i+k-1, m\}$. For every $1 \le i \le m$, define $\delta_i \equiv \{e \in E : l_e = i\}$ and $\Delta_i \equiv \{e \in E : i \in P(e)\}$. That is, $\Delta_i$ is the set of edges for which position $i$ is allowed.

Trying all possible Eulerian paths in $D$ and testing for each one whether it complies with $P$ might take exponential (in $m$) time. Instead, we iteratively construct, for every $i = 1, \ldots, m+1$, a list $\Phi_i$ of pairs $(v, S)$, such that $v$ is the last vertex of some path of length $i-1$ and $S$ is the list of edges that may extend the path from $v$. The algorithm is summarized below:

Compute $\delta_i$ for all $i$; **If** for some $i$, $|\delta_i| > k$ **then** return *False*.
Initialize $\Phi_1 \leftarrow \{(v, \delta_1) : \exists w, (v,w) \in \delta_1\}$, $\Phi_{m+1} \leftarrow \emptyset$, $\Delta_1 \leftarrow \delta_1$, $i \leftarrow 1$.
**While** $\Phi_i \neq \emptyset$ and $i \le m$ **do**:
  $i \leftarrow i+1$.
  $\Delta_i \leftarrow \Delta_{i-1} \cup \delta_i \backslash \delta_{\underline{i}-1}$; **If** $|\Delta_i| \ge 2k$ **then** return *False*.
  $\Phi_i \leftarrow \{(w, S \cup \delta_i \backslash \{(v,w)\}) : (v,S) \in \Phi_{i-1}, (v,w) \in S, S \cap \delta_{\underline{i}-1} \subseteq \{(v,w)\}\}$.
**If** $\Phi_{m+1} \neq \emptyset$ **then** return *True*; **Else** return *False*.

The following lemma, which is mentioned by Hannenhalli *et al.* (1996), is crucial for the analysis of the algorithm. The subsequent theorem proves the correctness of the algorithm.

**Lemma 8.** *If (D, P) has a compliant Eulerian path, then* $|\Delta_i| \le 2k-1$ *for all i.*

**Proof.** Let $\pi$ be an Eulerian path compliant with $(D,P)$. Then $\Delta_i \subseteq \{\pi(\underline{i}), \ldots, \pi(\overline{i})\}$ for all $i$. ■

**Theorem 9.** *The algorithm returns* True *if and only if there is an Eulerian path compliant with* $(D,P)$.

**Proof.** Call a pair $(w,S)$ *i-valid* if there exists a path $\{w_1, \ldots, w_i = w\}$ in $D$ such that: (1) $j \in P((w_j, w_{j+1}))$ for all $1 \le j < i$; (2) $S = \Delta_i \backslash \{(w_1,w_2), \ldots, (w_{i-1}, w_i)\}$; and (3) $\bigcup_{j=1}^{i-1} \delta_j \subseteq \{(w_1,w_2), \ldots, (w_{i-1}, w_i)\}$. Intuitively, (1) ensures that all edges in the path occupy allowed positions,

(2) ensures that the next ($i$-th) edge is both allowed for position $i$ and was not used already, and finally, (3) ensures that any edge that had to be used before position $i$ was indeed used. By induction on $i$, one can show that $\Phi_i$ is the set of all $i$-valid pairs. It follows that $(v_0, \ldots, v_m)$ is an Eulerian path compliant with $(D, P)$ if and only if $(v_m, \emptyset)$ is an $(m+1)$-valid pair. ■

We now analyze the complexity of the algorithm. Define $\Psi_i \equiv \{S : \exists w, (w, S) \in \Phi_i\}$ and $V_i \equiv \{v : \exists S, (v, S) \in \Phi_i\}$. A simple induction argument establishes the following lemma:

**Lemma 10.** *For each $i$ there exists a constant $s_i$, such that all sets $S \in \Psi_i$ satisfy $|S| = s_i$.*

**Corollary 3.** *For all $i$, $|\Psi_i| = O\left(\frac{4^k}{\sqrt{k}}\right)$.*

**Proof.** Every set $S \in \Psi_i$ satisfies $S \subseteq \Delta_i$. Hence, using Lemma 8 $|\Psi_i| \leq \binom{|\Delta_i|}{s_i} \leq \binom{2k-1}{k} = O\left(\frac{4^k}{\sqrt{k}}\right)$. ■

**Theorem 11.** *The algorithm can be implemented in $O(mk^{1.5}4^k)$ time and $O(m + k^{1.5}4^k)$ space.*

**Proof.** Both time and space bottlenecks involve computing the sets $\Phi_i$. The above bounds are attained by using the following data structure: For each $v \in V_i$, we keep a linked list $L_i(v)$ containing all sets $S \in \Psi_i$ such that $(v, S) \in \Phi_i$. We represent a subset $S \subseteq \Delta_i$ by a $(2k-1)$-bit vector, whose bits correspond to $\Delta_i$ edges. Since $|L_i(v)| = O(|\Psi_i|)$ for each $v \in V_i$, $\Phi_{i+1}$ can be computed from $\Phi_i$ in $O(|\Delta_i| \cdot |\Psi_i| \cdot k)$ time and space. By Lemma 8 and Corollary 3, this amounts to $O(k^{1.5}4^k)$ time (per iteration) and space. Management of the data structure requires additional $O(m)$ space. ■

We note that the algorithm can be modified to generate also one (or all) of the compliant Eulerian paths, e.g., by keeping a record of the last edge(s) leading to each $i$-valid pair. We also note that our algorithm can be extended to handle the PEP problem even when the sets of allowed positions for each edge need only be subsets of fixed-length intervals.

## ACKNOWLEDGMENTS

A. Ben-Dor was supported by the Program for Mathematics and Molecular Biology. I. Pe'er was supported by the Clore Foundation scholarship. R. Shamir was supported in part by a grant from the Ministry of Science, Israel. R. Sharan was supported by an Eshkol fellowship from the Ministry of Science, Israel.

## REFERENCES

Adleman, L.M. 1998. Location sensitive sequencing of DNA. Technical report, University of Southern California.

Apsvall, B., Plass, M.F., and Tarjan, R. 1979. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters* 8(3), 121–123.

Arratia, R., Martin, D., Reinert, G., and Waterman, M.S. 1997. Poisson process approximation for sequence repeats, and sequencing by hybridization. *J. Comp. Biol.* 3(3), 425–463.

Bains, W., and Smith, G.C. 1988. A novel method for nucleic acid sequence determination. *J. Theor. Biol.* 135, 303–307.

Ben-Dor, A., Pe'er, I., Shamir, R., and Sharan, R. 1999. On the complexity of positional sequencing by hybridization, *in* Crochemore, M., and Paterson, M., eds. *Proc. 10th Annual Symposium on Combinatorial Pattern Matching*, vol. 1645 of *Lecture Notes in Computer Science*, 88–100, Springer-Verlag, Berlin.

Ben-Dor, A., Pe'er, I., Shamir, R., and Sharan, R. 2001. Positional sequencing by hybridization. The Electronic Colloquium on Computational Complexity, TR01-054, July 2001.

Broude, S.D., Sano, T., Smith, C.S., and Cantor, C.R. 1994. Enhanced DNA sequencing by hybridization. *Proc. Natl. Acad. Sci. USA* 91, 3072–3076.

Drmanac, R., and Crkvenjakov, R. (1987). Yugoslav Patent Application 570.

Dyer, M., Frieze, A., and Suen, S. 1994. The probability of unique solution of sequencing by hybridization. *J. Comp. Biol.* 1, 105–110.

Frieze, A., and Halldorsson, B. 2001. Optimal sequencing by hybridization, *in Proc. 5th Annual Int. Conf. on Computational Molecular Biology (RECOMB'01)*, 141–148.

Gusfield, D., Karp, R., Wang, L., and Stelling, P. 1998. Graph traversals, genes and matroids: An efficient case of the travelling salesman problem. *Discrete Applied Mathematics* 88, 167–180.

Hannenhalli, S., Pevzner, P., Lewis, H., and Skiena, S. 1996. Positional sequencing by hybridization. *Computer Applic. Biosci.* 12, 19–24.

Khrapko, K.R., Lysov, Y.P., Khorlyn, A.A., Shick, V.V., Florentiev, V.L., and Mirzabekov, A.D. 1989. An oligonucleotide hybridization approach to DNA sequencing. *FEBS Letters* 256, 118–122.

Lysov, Y., Floretiev, V., Khorlyn, A., Khrapko, K., Shick, V., and Mirzabekov, A. 1988. DNA sequencing by hybridization with oligonucleotides. *Dokl. Acad. Sci. USSR* 303, 1508–1511.

Macevics, S.C. 1989. International Patent Application PS US89 04741.

Pe'er, I., and Shamir, R. 2000. Spectrum alignment: Efficient resequencing by hybridization, *in Proc. 8th Int. Conf. on Intelligent Systems in Molecular Biology (ISMB'00)*, 260–268.

Pevzner, P.A. 1989. l-tuple DNA sequencing: Computer anaylsis. *J. Biomol. Struct. Dyn.* 7, 63–73.

Pevzner, P.A., and Lipshutz, R.J. 1994. Towards DNA sequencing chips, *in Symposium on Mathematical Foundations of Computer Science*, 143–158. Springer, LNCS vol. 841.

Pevzner, P.A., Lysov, Y.P., Khrapko, K.R., Belyavsky, A.V., Florentiev, V.L., and Mirzabekov, A.D. 1991. Improved chips for sequencing by hybridization. *J. Biomol. Struct. Dyn.* 9, 399–410.

Preparata, F., Frieze, A., and Upfal, E. 1999. On the power of universal bases in sequencing by hybridization, *in Proc. 3rd Annual Int. Conf. on Computational Molecular Biology (RECOMB'99)*, 295–301.

Shamir, R., and Tsur, D. 2001. Large scale sequencing by hybridization, *in Proc. 5th Annual Int. Conf. on Computational Molecular Biology (RECOMB'01)*, 269–277.

Skiena, S.S., and Sundaram, G. 1995. Reconstructing strings from substrings. *J. Comput. Biol.* 2, 333–353.

Southern, E. 1988. UK Patent Application GB8810400.

Southern, E.M. 1996. DNA chips: Analysing sequence by hybridization to oligonucleotides on a large scale. *Trends in Genetics* 12, 110–115.

Southern, E.M., Maskos, U., and Elder, J.K. 1992. Analyzing and comparing nucleic acid sequences by hybridization to arrays of oligonucleotides: Evaluation using experimental models. *Genomics* 13, 1008–1017.

Address correspondence to:
*Itsik Pe'er*
*School of Computer Science*
*Lebanon Street*
*Tel Aviv University*
*Tel Aviv 69978, Israel*

*E-mail:* izik@tau.ac.il