

The Enhanced Double Digest Problem for DNA Physical Mapping

Ming-Yang Kao* Jared Samet† Wing-Kin Sung‡

October 31, 2018

Abstract

The *double digest problem* is a common NP-hard approach to constructing physical maps of DNA sequences. This paper presents a new approach called the *enhanced double digest problem*. Although this new problem is also NP-hard, it can be solved in linear time in certain theoretically interesting cases.

Key words. DNA physical mapping, fast algorithms, graph-theoretic techniques, NP-hardness

1 Introduction

The physical mapping of DNA is a key problem in computational biology [4]. A *map* of a DNA sequence consists of the locations of some given small sequences like e.g. GAATTC. Biologists use such maps in a preparatory step to determine the target DNA sequence [5].

A common technique of constructing maps uses restriction enzymes to cut a DNA sequence at the positions where a particular short DNA sequence appears. These positions are called *restriction sites*. One approach to modeling map construction is the *double digest* (DD) problem. Given two restriction enzymes \mathcal{A} and \mathcal{B} , this approach cuts a target DNA sequence using enzyme \mathcal{A} , enzyme \mathcal{B} , and both enzymes, separately. It is a biology fact that the restriction sites for enzymes \mathcal{A} and \mathcal{B} do not coincide. Throughout this paper, we make use of this fact. Let A , B and C be the three multisets of the lengths of the fragments formed after applying enzyme \mathcal{A} , enzyme \mathcal{B} and both enzymes to the target DNA sequence, respectively. Given A , B and C , the DD problem asks for permutations of the lengths in A and B such that if these sets of lengths are plotted on top of one another, the lengths of all the resulting subintervals formed due to overlapping match exactly the lengths in C . See Figure 1 for an example.

*Department of Computer Science, Yale University, New Haven, CT 06520, USA. Email: kao-ming-yang@cs.yale.edu. Research supported in part by NSF Grant 9531028.

†Yale College, New Haven, CT 06520, USA. Email: jared.samet@yale.edu.

‡Department of Computer Science, Yale University, New Haven, CT 06520, USA. Email: sung-ken@cs.yale.edu.

(a)	17	37	12	15	9	
(b)	46		38		6	
(c)	17	29	8	12	15	3

Figure 1: Stripes (a), (b) and (c) show the fragments resulting from the applications of enzyme \mathcal{A} , enzyme \mathcal{B} and both enzymes, respectively. In strip (c), the subfragments are created due to the overlapping between fragments in (a) and those in (b).

Many algorithms [6–8, 10] have been proposed for the DD problem. Stefik [9] gave the first algorithm using artificial intelligence. Fitch, Smith and Ralph [1] reduced the DD problem to the set partition problem. Goldstein and Waterman [3] approached this problem with a stochastic annealing heuristic for the traveling salesman problem. They also showed that the DD problem is NP-hard by reducing the set partition problem to it.

This paper suggests a new approach, called the *enhanced double digest* (EDD) problem. The EDD problem uses A , B , C and some additional length information; see Section 2 for the details of the approach. Although the EDD problem is still NP-hard, we show that if the lengths in C are all distinct, it can be solved in linear time. We also generalize the algorithm for the case where the number of duplicates in C is bounded by a constant. The time complexity of this generalized algorithm remains linear.

Section 2 details the new approach to define the EDD problem formally. Section 3 gives the linear-time algorithm for the case where C is duplicate-free. Also, it generalizes the algorithm to handle a small number of duplicate lengths. Section 4 proves that the EDD problem is NP-hard. Section 5 concludes with some directions for further work.

2 Problem formulation

Consider a target DNA sequence and two restriction enzymes \mathcal{A} and \mathcal{B} .

- By applying enzyme \mathcal{A} (respectively, \mathcal{B}) to the target DNA sequence, we obtain p (respectively, q) fragments. Let $A = \{a_1, \dots, a_p\}$ (respectively, $B = \{b_1, \dots, b_q\}$) be the multiset of the lengths of these p (respectively, q) fragments.
- For $i = 1, \dots, p$, let \hat{a}_i be the fragment corresponding to a_i . We apply enzyme \mathcal{B} to the fragment \hat{a}_i and obtain a set of subfragments. Let AB_i be the multiset of the lengths of these subfragments.
- For $j = 1, \dots, q$, let \hat{b}_j be the fragment corresponding to b_j . We apply enzyme \mathcal{A} to the fragment \hat{b}_j and obtain a set of subfragments. Let BA_j be the multiset of the lengths of these subfragments.

For the example in Figure 1, the following length information is gathered:

- $A = \{a_1 = 9, a_2 = 12, a_3 = 15, a_4 = 17, a_5 = 37\}; B = \{b_1 = 6, b_2 = 38, b_3 = 46\};$
- $AB_1 = \{3, 6\}; AB_2 = \{12\}; AB_3 = \{15\}; AB_4 = \{17\}; AB_5 = \{8, 29\};$
- $BA_1 = \{6\}; BA_2 = \{3, 8, 12, 15\}; BA_3 = \{17, 29\}.$

It is easily verified that the data found in this way has the following properties:

Fact 1.

1. For $i = 1, \dots, p$, $a_i = \sum_{c \in AB_i} c$. For $j = 1, \dots, q$, $b_j = \sum_{c \in BA_j} c$.
2. $\cup_i AB_i = \cup_j BA_j = C$.
3. $|C| = |A| + |B| - 1$.

Proof. Straightforward. □

Given $A, B, AB_1, \dots, AB_p, BA_1, \dots, BA_q$, the *enhanced double digest problem* \mathcal{P} asks for a *valid permutation* (π_A, π_B) of the elements in A and B such that the following can be achieved. When the fragments \hat{a}_i for $a_i \in A$ and \hat{b}_j for $b_j \in B$ are plotted on the same line according to the order given by π_A and π_B , a set of subfragments is formed due to overlapping. The multiset C of the lengths of these subfragments is required to be equal to $\cup_{i=1}^p AB_i = \cup_{j=1}^q BA_j$. In addition,

- for every $a_i \in A$ (respectively, $b_j \in B$), AB_i (respectively, BA_j) is equal to the multiset of the lengths of the subfragments which overlap with \hat{a}_i (respectively, \hat{b}_j).

Note that an instance of this problem may have no solution or more than one valid permutation. The algorithms given in Section 3 can recover all valid permutations, if any exists.

3 An efficient algorithm

Unless otherwise stated, this section assumes that C has no duplicates. Let $n = |C|$. This section shows that the EDD problem \mathcal{P} can be solved in $O(n)$ time.

Section 3.1 formulates the EDD problem as a graph problem. Section 3.2 describes the linear-time algorithm. Section 3.3 discusses how to generalize this linear-time algorithm to the case where C may contain a small number of duplicates.

3.1 A graph representation

Given $A, B, AB_1, \dots, AB_p, BA_1, \dots, BA_q$, we construct an undirected graph G as follows.

- The node set of $G = A \cup B \cup C$.
- For every $a_i \in A$ and every $x \in C$, $(a_i, x) \in G$ if $x \in AB_i$.

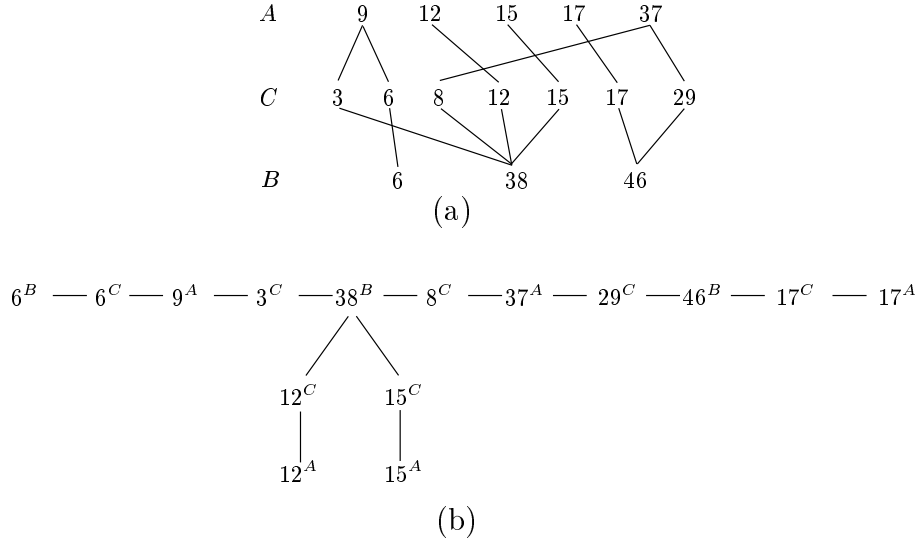


Figure 2: The graph G in (a) is constructed from the example in Figure 1. G can be redrawn into a tree as shown in (b). The superscript A, B or C of each node denotes whether the node belongs to A, B or C .

- For every $b_j \in B$ and every $x \in C$, $(b_j, x) \in G$ if $x \in BA_j$.

From the definition, we can observe that G satisfies the following lemma.

Lemma 2. *G is connected. For each node in $A \cup B$, its degree is at least 1 and it is adjacent to nodes in C only. Also, every node in C connects to exactly one node in A and one node in B .*

Proof. Straightforward based on the assumption that C has no duplicates. \square

If \mathcal{P} has a valid permutation, G has two more properties as stated in Lemma 3. Figure 2 illustrates an example. A *diameter* of a tree is a path with the largest number of edges. A *dangler* is a 2-node-long path. Given a tree T , a subtree τ of T is said to be *hanged on* a path P in T if τ is a tree in the forest $T - P$.

Lemma 3. *If \mathcal{P} has a valid permutation, then the following statements hold.*

1. G is a tree.
2. For any diameter S of G , the subtrees hanged on S must be danglers.

Proof.

Statement 1. To prove by contradiction, suppose that G contains a cycle D . By the construction of G , D must be of the form

$$a_{i_1}, c_{k_1}, b_{j_1}, c_{k_2}, a_{i_2}, c_{k_3}, b_{j_2}, c_{k_4}, \dots, c_{k_{2z}}, a_{i_{z+1}},$$

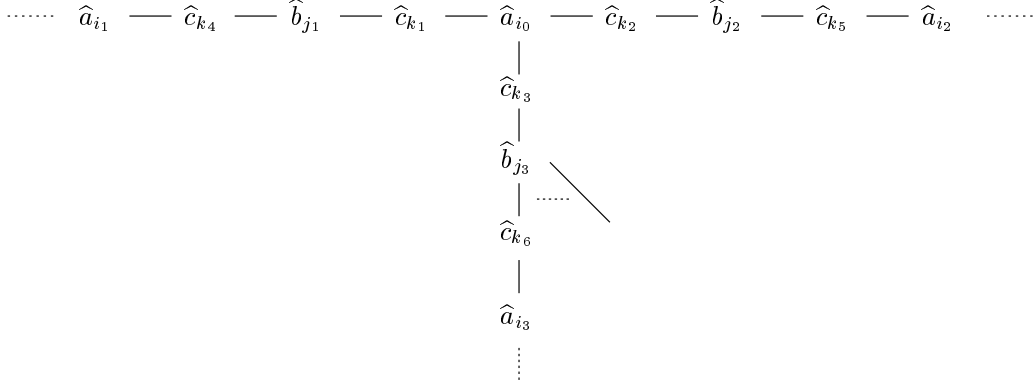


Figure 3: In this example, all $a_i \in A$, $b_j \in B$ and $c_k \in C$.

where $i_1 = i_{z+1}$; $a_{i_1}, \dots, a_{i_z} \in A$; $b_{j_1}, \dots, b_{j_z} \in B$; and $c_{k_1}, \dots, c_{k_{2z}} \in C$.

By definition, if a_i, c_k, b_j is a path in G , then \hat{a}_i and \hat{b}_j overlap by \hat{c}_k in any valid permutation of \mathcal{P} . Thus, for $1 \leq \ell \leq z-1$, the existence of the subpath $a_{i_\ell}, \dots, a_{i_{\ell+2}}$ of D in G means that \hat{b}_{i_ℓ} overlaps with \hat{a}_{i_ℓ} and $\hat{a}_{i_{\ell+1}}$ and $\hat{b}_{i_{\ell+1}}$ overlaps with $\hat{a}_{i_{\ell+1}}$ and $\hat{a}_{i_{\ell+2}}$. To enable both \hat{b}_{i_ℓ} and $\hat{b}_{i_{\ell+1}}$ overlap with $\hat{a}_{i_{\ell+1}}$, $\hat{a}_{i_{\ell+1}}$ must be in the middle of \hat{a}_{i_ℓ} and $\hat{a}_{i_{\ell+2}}$ for $1 \leq \ell \leq z-1$. Consequently, for $1 \leq \ell \leq z-1$, \hat{a}_{i_ℓ} is in the middle of \hat{a}_{i_1} and $\hat{a}_{i_{z+1}} = \hat{a}_{i_1}$, which is impossible.

Statement 2. For any diameter S of G , we show that every subtree τ hanged on S must be a dangler. First, τ must be hanged on S at a node in $A \cup B$. Otherwise, if τ is hanged on S at a node $c \in C$, c has degree greater than 2, contradicting Lemma 2. Then, τ has more than one node because the root of τ is a node in C and must be of degree 2. If τ cannot have more than 2 nodes, Statement 2 follows.

To prove by contradiction, suppose that τ has more than two nodes. Without loss of generality, assume that τ is hanged on S at a node $a_{i_0} \in A$ and the root of τ is a node $c_{k_3} \in C$. Note that c_{k_3} has another neighbour, say b_{j_3} , from B . If τ contains more than two nodes, b_{j_3} must have a child, say c_{k_6} , from C and c_{k_6} must have a child, say a_{k_3} , from A . Thus, τ must have a root-to-leaf path of length more than 4. Then, the two paths from a_{i_0} to both ends of S must be of length more than 4. Otherwise, S cannot be a diameter of G . From those observations, G has the pattern shown in Figure 3. According to the pattern, $\hat{b}_{j_1}, \hat{b}_{j_2}$ and \hat{b}_{j_3} overlap with \hat{a}_{i_0} . Therefore, in any valid permutation, one of $\hat{b}_{j_1}, \hat{b}_{j_2}$ and \hat{b}_{j_3} , say \hat{b}_{j_2} , must be in the middle of the other two fragments and \hat{b}_{j_2} can only overlap with \hat{a}_{i_0} . However, according to the pattern in Figure 3, for $\ell = 1, 2, 3$, \hat{b}_{j_ℓ} overlaps with another fragment \hat{a}_{i_ℓ} , reaching a contradiction. \square

Now, we know that if \mathcal{P} has a valid permutation, G satisfies the two properties of Lemma 3. The remainder of this section show that the converse of this statement is also true. Suppose that G is a tree with a diameter S such that all the subtrees hanged on S are danglers. We define π_C to be a permutation on C formed by a search defined below.

Dangler-first search: Traverse G starting from one end of S to the other end of S ; read

off the nodes in C on S ; whenever meet any node x with degree greater than 2, read off the nodes in C in the dangles hanged on S at x in any order and continue to traverse S .

Lemma 4. *The elements in each AB_i form a consecutive subsequence in π_C . Similarly, the elements in each BA_j form a consecutive subsequence in π_C .*

Proof. For each i , if AB_i contains only one element, then the lemma follows. Otherwise, a_i is of degree at least 2. Then, a_i must be on the diameter S . Let c and c' be elements in AB_i which are the two neighbours of a_i on S . The remaining nodes in AB_i must be located in the dangles hanged on S at a_i . By dangle-first search, all the elements in AB_i must form a consecutive subsequence in π_C . By symmetry, for each j , the elements in BA_j must form a consecutive subsequence in π_C . \square

By Lemma 4, π_C can be partitioned into p subintervals such that the r th interval contains the elements in AB_{i_r} for $r = 1, \dots, p$. Let π_A be the permutation $(a_{i_1}, \dots, a_{i_p})$. Similarly, π_C can be partitioned into q intervals such that the s th interval contains the elements in BA_{j_s} for $s = 1, \dots, q$. Let π_B be the permutation $(b_{j_1}, \dots, b_{j_q})$. We call (π_A, π_B) the *induced permutation* of π_C .

Lemma 5. *The induced permutation (π_A, π_B) of π_C is a valid permutation of \mathcal{P} .*

Proof. Suppose the lengths from A, B and C are plotted on the same line according to the order given by π_A, π_B and π_C , respectively. Consider the stripes formed from A and C . By Fact 1 and Lemma 4, for each i , \hat{a}_i overlaps with \hat{c} for all $c \in AB_i$. By symmetry, for each j , \hat{b}_j overlaps with \hat{c} for all $c \in BA_j$. Then, by the definition of the EDD problem, (π_A, π_B) is a valid permutation. \square

Theorem 6. *Given the enhanced double digest problem \mathcal{P} and its corresponding graph G , \mathcal{P} has a valid permutation if and only if G satisfies the two properties in Lemma 3.*

Proof. The only-if part follows from Lemma 3. The if part follows from Lemma 5. \square

3.2 A linear-time algorithm for a duplicate-free C

This section describes how to compute a valid permutation of \mathcal{P} in $O(n)$ time. The algorithm is as follows.

Algorithm Enhanced-Double-Digest

1. Construct the graph G corresponding to \mathcal{P} .
2. If G does not satisfy the two properties in Lemma 3, then return “no valid permutation”.
3. Find the permutation π_C using dangle-first search.
4. Find the induced permutation (π_A, π_B) of π_C .
5. Return (π_A, π_B) .

Lemma 7. *Algorithm Enhanced-Double-Digest can correctly find a valid permutation in $O(n)$ time.*

Proof. First, by Lemma 5 and Theorem 6, Enhanced-Double-Digest is correct. As for its time complexity, Step 1 requires $O(n)$ time as G contains $2n$ edges and we can find each edge in $O(1)$ time. Step 2 checks whether G satisfies the two properties in Lemma 3. For property 1, we can determine whether a graph is a tree in $O(n)$ time. For property 2, we can compute a diameter of a tree in linear time first, then, we verify whether G satisfies property 2 by detecting whether the subtrees hanged on the diameter are dangles. Thus, Step 2 requires $O(n)$ time. Step 3 finds π_C using dangle-first search. Since the search scans every node in G once, it runs in $O(n)$ time. Step 4 finds the induced permutation (π_A, π_B) of π_C in $O(n)$ time. In summary, a valid permutation of \mathcal{P} can be computed in $O(n)$ time. \square

By modifying Algorithm Enhanced-Double-Digest slightly, we can report all valid permutations of \mathcal{P} . First, observe that the valid permutations of \mathcal{P} depend on the possible permutations π_C . There are three cases.

Case 1: G does not have any dangle. Then, there is a unique π_C . Thus, the current algorithm reports all valid permutations of \mathcal{P} .

Case 2: G has one set of dangles hanged on one node of its diameter. Then, the possible permutations π_C depend on the permutation of the set of nodes in the dangles which belong to C . For the example in Figure 2, the possible permutations π_C can be represented by

$$6, 3, \text{permutation}(12, 15), 8, 29, 17.$$

All valid permutations π_A and π_B can be represented by 9, permutation(12, 15), 37, 17 and 6, 38, 46, respectively. These valid permutations can be reported by modifying Steps 3 and 4 of the algorithm. The time complexity of the modified algorithm is still $O(n)$.

Case 3: G has k sets of dangles hanged on k respective nodes of its diameter. Then, the possible permutations π_C can be represented similarly, except that each π_C contains k permutation blocks. The above modified algorithm is sufficient to report all valid permutations of \mathcal{P} .

3.3 A general algorithm for C with few duplicates

The algorithm Enhanced-Double-Digest in Section 3.2 can solve the EDD problem if C contains no duplicates. Here, we give an algorithm which works without this assumption.

First, we consider the following example.

- $A = \{a_1 = 18, a_2 = 19\}; B = \{b_1 = 4, b_2 = 5, b_3 = 7, b_4 = 8, b_5 = 13\};$
- $AB_1 = \{5, 6, 7\}; AB_2 = \{4, 7, 8\};$
- $BA_1 = \{4\}; BA_2 = \{5\}; BA_3 = \{7\}; BA_4 = \{8\}; BA_5 = \{6, 7\}.$

In this example, there are two 7's in $C = \cup_i AB_i = \cup_j BA_j$. These two 7's in fact represent two different subfragments in the target DNA sequence. To distinguish them, let the copy of 7 in AB_1 be 7_1 and that in AB_2 be 7_2 . Since 7 also belongs to BA_3

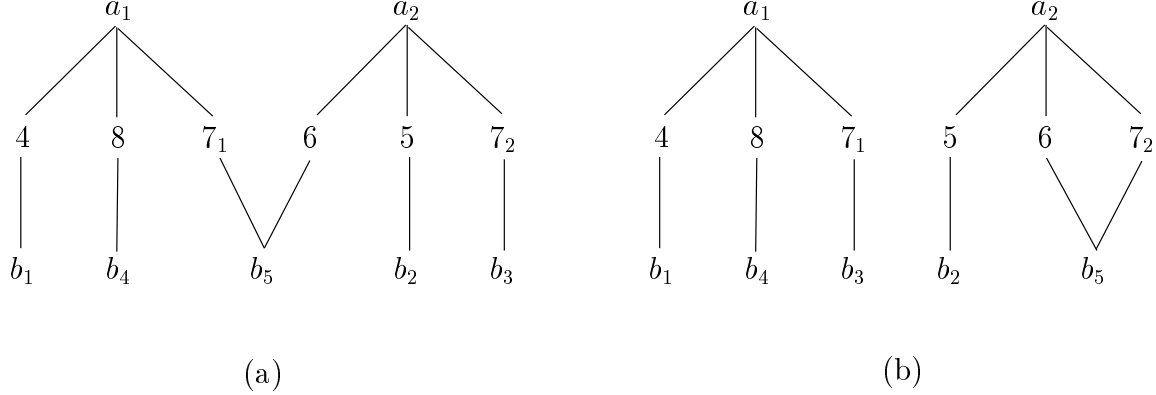


Figure 4: (a) is the case where $7_1 \in BA_5$ and $7_2 \in BA_3$; (b) is the case where $7_1 \in BA_3$ and $7_2 \in BA_5$.

and BA_5 , there are two possible combinations, namely, (a) $7_1 \in BA_5$ and $7_2 \in BA_3$ and (b) $7_1 \in BA_3$ and $7_2 \in BA_5$. Figure 4(a) and 4(b) illustrate the graph G for both cases; from these two graphs G , we can obtain a valid permutation from combination (a). Therefore, we can handle duplicates in C by giving them different subscripts. Then, all the elements in C are different and we can solve the enhanced double digest problem using the algorithm Enhanced-Double-Digest in Section 3.2. More precisely, we have the following algorithm.

1. If C contains duplicates, then we assign a unique subscript to each duplicate.
2. For each possible combinations of the subscripts in the duplicates, we execute Enhanced-Double-Digest to compute a valid permutation.

Let ℓ be the number of duplicates in C . The above algorithm execute Enhanced-Double-Digest for at most $\ell!$ time. Therefore, a valid permutation can be computed in $O(\ell!n)$ time. Thus, if ℓ is constant, the generalized algorithm still runs in linear time.

4 The enhanced double digest problem is NP-hard

This section proves the NP-hardness of the enhanced double digest problem by a reduction from the Hamiltonian Path problem [2].

Given an undirected graph H , we show that in polynomial time, we can construct an EDD instance \mathcal{Q} so that H contains a hamiltonian path if and only if \mathcal{Q} has a valid permutation. For ease of prove, we augment H with two new nodes t and z . All nodes originally in H have edges to t . In addition, we add an edge (t, z) to H . Note that the original H contains a hamiltonian path if and only if the amended H has a hamiltonian path. Let ℓ be the number of nodes in H . Assume that the nodes in H are labeled by $\{1, 2, \dots, \ell\}$. For each node v , let $\kappa(v)$ be the number of neighbours of v . Let $v' = v + \ell$.

The EDD instance \mathcal{Q} is given the following length information. Note that this length information can be constructed from H in polynomial time.

- $A = \{a_v \mid v \in H\}$ where $a_z = t'$, $a_t = t + \sum_{u \in H - \{t, z\}} u'$ and $a_v = v + \sum_{(u, v) \in H} u'$ for $v \neq z, t$. Also, $AB_z = \{t'\}$; $AB_t = \{u' \mid u \in H - \{t, z\}\} \cup \{t\}$; and $AB_v = \{u' \mid (u, v) \in H\} \cup \{v\}$ for $v \neq z$.
- $B = \{b_v, b_{v(1)}, \dots, b_{v(\kappa(v)-1)} \mid v \in H - \{z\}\}$ where $b_v = v + v'$ and $b_{v(i)} = v'$ for all $v \in H - \{z\}$ and all $i \leq \kappa(v) - 1$. Also, $BA_v = \{v, v'\}$ and $BA_{v(i)} = \{v'\}$.

Lemma 8. H has a hamiltonian path if and only if there is a valid permutation for \mathcal{Q} .

Proof. The two directions are proved as follows.

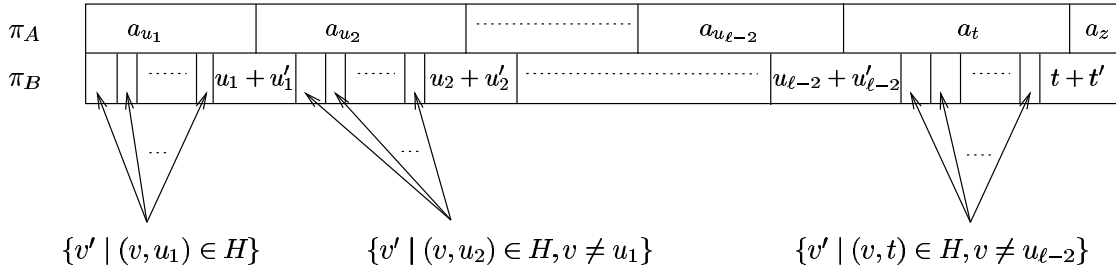


Figure 5: The permutations π_A and π_B of A and B , respectively.

(\implies) Let $u_1, u_2, \dots, u_{\ell-2}, t, z$ be a hamiltonian path in H . Let π_A and π_B be permutations of A and B as shown in Figure 5. It is easy to check that (π_A, π_B) is a valid permutation to \mathcal{Q} .

(\impliedby) Let (π_A, π_B) be a valid permutation of \mathcal{Q} . The remainder of this proof shows that the ordering of the lengths in π_A defines a hamiltonian path in H .

Assume the lengths from A are plotted on a line according to the order given by π_A and similarly, the lengths from B are also plotted on this line according to π_B . For each $v \in H$, the line fragment corresponds to $a_v \in A$ is called \hat{a}_v . For each $v \in H - \{z\}$, the line fragment corresponds to $b_v \in B$, is called \hat{b}_v .

For every $v \in H - \{z\}$, since $BA_v = \{v, v'\}$, \hat{b}_v overlaps with two consecutive line fragments from A ; in addition, the overlapping regions between \hat{b}_v and these two line fragments must be of length v and v' , respectively. Observe that $v \in AB_v$ and $v \notin AB_u$ for all $u \neq v$. One of these two fragments, which overlaps with \hat{b}_v , must be \hat{a}_v . The other line fragment can be \hat{a}_u for any $u \in H$ with $v' \in AB_u$, i.e., $(v, u) \in H$.

Let $\pi_A = (a_{u_1}, \dots, a_{u_\ell})$. From the above argument, we know that, for every two consecutive line fragments \hat{a}_i and \hat{a}_{i+1} , there exists a fragment \hat{b}_v (where v is either u_i or u_{i+1}) which overlaps with both \hat{a}_{u_i} and $\hat{a}_{u_{i+1}}$. The above argument also implies that $(u_i, u_{i+1}) \in H$. Thus, u_1, \dots, u_ℓ forms a path in H . As u_1, \dots, u_ℓ contains all the ℓ nodes of H , this path is a hamiltonian path. \square

5 Further research directions

This highly theoretical work can be extended in several directions. One direction is to design a series of laboratory procedures that can actually produce the input length information in the required form. Another direction is to consider the problem of more than 2 digesting enzymes. Using multiple enzymes could help resolve the issue of multiple solutions that arise when there are dangles or duplicate subfragment lengths. Also, the extra input may actually make the problem solvable in a shorter period of time. The third direction is to have a probabilistic analysis of the number of duplicates in C , when the length of the target DNA sequence is given. It would be the most meaningful to conduct such analysis under a probabilistic model that is derived specifically for feasible laboratory procedures. Lastly, this paper does not address the issue of noise in the length data. From the practical point of view, handling noise effectively is very important.

6 Acknowledgments

We wish to thank the anonymous referees for many helpful suggestions.

References

- [1] W. M. Fitch, T. F. Smith, and W. W. Ralph. Mapping the order of DNA restriction fragments. *Gene*, 22:19–29, 1983.
- [2] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, NY, 1979.
- [3] L. Goldstein and M. S. Waterman. Mapping DNA by stochastic relaxation. *Advances in Applied Mathematics*, 8:194–207, 1987.
- [4] R. Karp. Mapping of the genome: Some combinatorial problems arising in molecular biology. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 278–285, 1993.
- [5] D. Nathans and H. O. Smith. Restriction endonucleases in the analysis and restructuring of DNA molecules. *Annual Review of Biochemistry*, 44:273–293, 1975.
- [6] P. A. Pevzner. DNA physical mapping, flows in networks and minimum cycles mean in graphs. In S. G. Gindikin, editor, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science 8: Mathematical Methods of Analysis of Biopolymer Sequences*, pages 99–112. American Mathematical Society, Providence, RI, 1992.
- [7] P. A. Pevzner. DNA physical mapping and alternating Eulerian cycles in colored graphs. *Algorithmica*, 13(1/2):77–105, 1995.

- [8] W. Schmitt and M. S. Waterman. Multiple solutions of DNA restriction mapping problems. *Advances in Applied Mathematics*, 12:412–427, 1991.
- [9] M. Stefik. Inferring DNA structure from segmentation data. *Artificial Intelligence*, 11:85–114, 1978.
- [10] M. S. Waterman and J. R. Griggs. Interval graphs and maps of DNA. *Bulletin of Mathematical Biology*, 48(2):189–195, 1986.