The Restriction Mapping Problem Revisited

Gopal Pandurangan [*]   H. Ramesh [†]

Abstract

In computational molecular biology, the aim of *restriction mapping* is to locate the restriction sites of a given enzyme on a DNA molecule. *Double digest* and *partial digest* are two well-studied techniques for restriction mapping. While double digest is NP-complete, there is no known polynomial algorithm for partial digest. Another disadvantage of the above techniques is that there can be multiple solutions for reconstruction.

In this paper we study a simple technique called *labeled partial digest* for restriction mapping. We give a fast polynomial time ($O(n^2 \log n)$ worst-case) algorithm for finding all the n sites of a DNA molecule using this technique. An important advantage of the algorithm is the *unique* reconstruction of the DNA molecule from the digest. The technique is also robust in handling errors in fragment lengths which arises in the laboratory. We give a robust $O(n^4)$ worst-case algorithm that can provably tolerate an absolute error of $O(\frac{\Delta}{n})$ (where Δ is the minimum inter-site distance), while giving a unique reconstruction. We test our theoretical results by simulating the performance of the algorithm on a real DNA molecule.

Motivated by the similarity to the labeled partial digest problem, we address a related problem of interest - the *de novo peptide sequencing* problem (Chen et.al, *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2000) which arises in the reconstruction of the peptide sequence of a protein molecule. We give a simple and efficient algorithm for the problem without using dynamic programming. The algorithm runs in time $O(k \log k)$, where k is the number of ions and is an improvement over the algorithm in Chen et.al.

[*]Department of Computer Science, Brown University, Providence, RI 02912, USA. email: `gopal@cs.brown.edu`

[†]Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India. email: `ramesh@csa.iisc.ernet.in`

1 Introduction

In computational molecular biology the aim of *restriction site mapping* is to locate the restriction sites of a given enzyme on a given DNA molecule (usually called the *target DNA*). Determining the location of sites from restriction site data is a difficult algorithmic problem and has been extensively studied [16, 6, 1]. See Pevzner's recent book ([10]) for an excellent introduction to restriction mapping, where many approaches are discussed.

In the well-known *partial digest* approach ([17, 15]) to restriction site mapping, we use a single enzyme to break up the DNA molecule into fragments of different lengths by exposing the enzyme for different time periods. The experiment provides data about *all* pairwise distances between restriction sites (the error-free case). Assuming there are n sites in the DNA molecule, (the error-free) partial digest approach is to reconstruct the DNA from the $\binom{n}{2}$ fragments formed by every two cuts. This problem is also called the *turnpike reconstruction problem*, where we have to reconstruct a set of n points (cities) on a line given the set of $\binom{n}{2}$ distances between them. It should be made clear that the correspondences between the distances and point pairs are *not* known and the entire difficulty of the reconstruction problem is to deduce such labeling information.

In restriction mapping, we have to deal with errors in fragment lengths which arises in the laboratory. A simple backtracking algorithm was proposed by Skiena et.al[14] whose running time can be exponential (in n, the number of sites) in the worst case. However, assuming that the fragment lengths are drawn from a certain probability distribution they showed that the algorithm runs in time $O(n^3)$ time with high probability while tolerating a relative error of $O(1/n^2)$ in fragment lengths.

A major disadvantage of the partial digest approach is the large number of possible solutions. Skiena et.al [14] show that the number of possible solutions is between $\frac{1}{2}n^{0.8107144}$ and $\frac{1}{2}n^{1.2324827}$ where n is the number of sites. In the presence of errors in fragment lengths, the number of possible solutions typically increases even more, posing a significant algorithmic problem. Hence, it is essential for a restriction mapping technique to have a very small number of solutions.

In this paper we study a new technique for constructing restriction maps which is a simple modification of partial digest. Our technique admits a fast polynomial time algorithm and gives a unique reconstruction, even under the presence of errors. In *labeled partial digest*, we will label both the ends of the DNA molecule by using radioactive labeling. Radio-labeling both ends of a double stranded DNA is a *standard* and easy laboratory technique in molecular biology. One standard method is radio-labeling of the 5' ends of the double stranded DNA with radioactive phosphate ($\gamma^{32}[P]$) [12].

We make a remark about the labeling of the ends of the DNA molecule. The reconstruction problem becomes trivial if we manage to label just one end of the DNA molecule. But labeling just one end has many laboratory difficulties associated with it, especially if we are finding restriction sites in large DNA molecules [12]. The difficulty of left-right orientation due to symmetry of a double stranded DNA [1] also arises in *optical mapping* [13] creating a basic combinatorial problem [8] .

We first show that the new approach (of labeling both ends) leads to a polynomial time algorithm for reconstruction from an error-free partial digest. An important advantage is

---

[1]Restriction site mapping is typically done with double stranded DNA as restriction enzymes can cut only such DNA as opposed to single stranded DNA.

that there is only a unique solution under this technique as opposed to exponential number in double digest and a polynomial number in partial digest. This gives confidence in reconstruction. Another advantage of this technique is that it is quite robust to errors in lengths, a common situation which arises in the laboratory.

Motivated by the similarity to the labeled partial digest problem, we also address a related problem of recent interest: the *de novo peptide sequencing problem* which arises in the problem of reconstructing the amino acid (or the peptide) sequence of a protein (Chen et.al [3]). We give a simple and efficient algorithm for the problem without using dynamic programming. Our algorithm, which is similar in spirit to our error-free labeled partial digest algorithm, runs in time $O(k \log k)$, where k is the number of ions and is an improvement over the best algorithm in Chen et.al which takes $O(k^2)$ time.

The paper is organized as follows. In section 2, we describe the partial digest technique and the labeling of the DNA molecule. We also briefly discuss other approaches to restriction mapping. In section 3, we describe a fast $((O(n^2 \log n)$ time) algorithm for (error-free) labeled partial digest and analyze its running time. In section 4, we analyze the robustness of the algorithm to errors in fragment lengths and show that it can tolerate an absolute error limit of $O(\frac{\Delta}{n})$, where Δ is the minimum inter-site distance. In section 5, we describe our results on a bacteriophage under simulated experimental conditions. In section 6, we give a simple and efficient algorithm for the *de novo peptide sequencing problem*. We conclude with open problems in section 7.

2   Background and Previous Work

Restriction mapping is a well studied problem in computational biology ([17, 10]). One of the first attempts was the method of *double digest* ([16]). However it suffers from serious computational problems. Goldstein and Waterman [7] showed that it is NP-complete. Further the number of possible solutions is exponential in the number of sites. *Partial digest* was proposed as an alternative technique, since the maximum number of solutions is smaller (only polynomial in the number of sites), from a combinatorial point of view it was thought to be easier. Nevertheless, it is not known whether partial digest is NP-complete or not. Other known approaches are *optical mapping* [13, 8] and the *probed partial digest mapping* [9]. See Pevzner [10] for a discussion of these approaches and the algorithmic problems associated with them.

We now discuss the partial digest approach in more detail. The laboratory procedure for partial digest is as follows [15]. We subject the target DNA to one restriction enzyme only (unlike double digest where we subject the DNA to *two* restriction enzymes), but perform many experiments on copies of the DNA, varying the time during which the enzyme acts on each copy. By giving more or less time to the enzyme, more or less restriction sites will be recognized, thus yielding fragments of different lengths. Fragment lengths are measured by a standard laboratory technique called *gel electrophoresis*. Ideally the experiments should provide us with at least one fragment for every pair of restriction sites. We then try to pinpoint the location of the restriction sites by analyzing fragment lengths. For example consider the target DNA with restriction sites for a particular enzyme as shown in figure 1. An *error-free* partial digest would result in the following fragment sizes: 3, 11, 17 and 27 (fragment length between the left endpoint and all other sites); 8, 14 and 24 (fragments between the first restriction site and those to its right); 6 and 16 (fragments between the
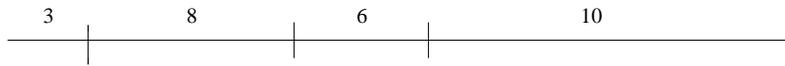
Figure 1: An instance for the partial digest problem

second site and those to its right) and 10 (last fragment).

Several kinds of experimental errors may occur with digestion data. A common error arises due to uncertainty in length measurement. The error can be up to 5% in gel electrophoresis [12, 10]. Another error is that some fragments may be missing because it is difficult to digest DNA in such a way that the cuts between every two sites are formed. These errors can substantially complicate the algorithmic problem. In this paper, we focus on handling errors that can arise from measuring fragment lengths.

We will now look at some of the algorithms known for partial digest. For the rest of the paper, we assume that the "parent" fragment has n restriction sites or markers. For simplicity we will assume both endpoints also to be counted as restriction sites.

Rosenblatt and Seymour [11] gave an elegant pseudo-polynomial algorithm for partial digest based on factorization of polynomials. However their algorithm does not generalize to noisy data. Later Skiena et.al [14] proposed a simple *backtracking* algorithm for the error-free partial digest problem. However this algorithm is *not* a polynomial time algorithm. Zhang [19] showed that it can take $\Theta(2^n n \log n)$ time in the worst case.

Skiena and Sundaram [15] analyzed the backtracking algorithm for the partial digest problem under the assumption that the location of the sites are chosen according to a binomial distribution. They showed that the algorithm runs in $O(n^3)$ with high probability. while tolerating a relative error $r = O(1/n^2)$ in the lengths of the fragments. Beyond that their method becomes infeasible.

Dakic [5] has recently proposed polynomial time solutions for certain classes of instances of the partial digest problem. These include the class of instances constructed by Zhang [19]. They also include the class of instances that have a unique solution *and* all the the $\binom{n}{2}$ distances are different and on which the backtracking algorithm backtracks only a constant number of steps. The technique used is writing a quadratic program for the problem and then doing a semidefinite relaxation.

In this paper we study a new technique called *labeled partial digest*. In this technique we label both ends of the double stranded DNA before we perform partial digest as outlined above. Labeling both ends of a double stranded DNA is a routine technique in the laboratory and is easily accomplished. One method is radio-labeling both the 5' ends by a radioactive phosphate ($\gamma^{32}[P]$). The 5' ends of the DNA molecule is dephosphorylated (the extra phosphate from the 5' ends are removed) with alkaline phosphatase. Then the DNA is incubated in the presence of T4 polynucleotide kinase and $\gamma[^{32}P]ATP$ [12]. After partial digest by the restriction enzyme we measure fragment lengths by gel electrophoresis as before. However, the labeling of ends enables us to identify fragments from *either* end of the DNA (which we call as *primary fragments*) .

In a recent work, Blazewicz et.al [2] study a variation of the partial digest method similar in certain aspects to our labeled partial digest approach. In their method, called the *simplified partial digest method*, the target DNA is divided into two sets. All molecules from the first set are cut in *at most one* site. This is done by properly chosen time spans allowed for the

reaction. Molecules from the other set are cut in *all* restriction sites by allowing the restriction enzyme to act for a long time. Then, the lengths of restriction fragments are measured. The experiment with the first set is equivalent to labeling - it enables us to identify the primary fragments. However, the second set identifies only those fragments *between consecutive* sites. The authors give an *exponential time* algorithm for the error-free version, based (essentially) on exhaustive search. However, the algorithm is shown to perform well in the average case. They also report simulation results on noisy data which is superior to those in [15]. The main advantage of their method, is its simplified experimental procedure. However, although their method (like ours) finds the primary fragments, it still suffers the same worst-case exponential complexity as partial digest. Our technique is different, as we use *all* the $\binom{n}{2}$ fragments from the partial digest, in addition to knowing the primary fragments. However, their work has two useful implications for our paper. First, their simpler methodology to find primary fragments can be used instead of labeling. Second, the advantages cited in handling experimental errors like missing fragments carries over to our method.

We will now introduce some terminology which will be used throughout the paper. We will call the endpoints of the parent DNA molecule as *end* sites and all the other sites as *intermediate*.

Assuming that we have an enzyme that can break the segment in all the sites, an error-free partial digest will produce $\binom{n}{2}$ fragments. The *restriction site* analysis involves reconstructing the restriction sites from the length of the fragments.

We will number the sites (points) as $1, \ldots, n$ starting from the left end, that is, site i is identified by x_i, the distance from the left end. We will assume that $x_1 = 0$ and $x_n = t$, where t is the length of the parent DNA molecule. Let the lengths of the fragments form the distance (multi-)set D and we will assume that $|D| = \binom{n}{2}$ in the error-free case. We refer to the fragments which have one endpoint as either the left end or the right end of the parent DNA molecule as *primary* fragments. All other fragments are called *secondary or intermediate* fragments. We note that there will be $2n-3$ primary fragments in the error-free case.

The backtracking algorithm of Skiena et.al ([14, 15]) is as follows. The basic idea in this algorithm is to choose in each "round" the largest remaining primary fragment and placing it first on the left end (note that there can be only be two choices for this fragment). Suppose we are in round i and let the largest remaining primary fragment be f_i. If the *derived* fragments (in round i, there will be $i - 1$ such fragment lengths, induced by f_i and already determined $i - 1$ sites) are in the distance set D then we continue to the next round. Otherwise we try to place f_i at the right end and try again. If we fail, then we backtrack to the previous round. We refer to Pevzner's book ([10]) for more details.

3   Labeled Partial Digest

We will consider the new approach where we label both ends of the DNA molecule as described. We do the partial digest in the laboratory on a number of such cloned DNA molecules. The effect of this is to produce $2n - 3$ labeled primary fragments, in addition to the secondary fragments. We will devise a new reconstruction algorithm which will make use of this additional information (i.e., the primary fragments). It is worth noting that this labeling information does not help in speeding up the greedy backtracking algorithm of Skiena et.al. This is easy to see because in each "round" of the greedy algorithm we still have

the problem of deciding the end for the current largest fragment, since primary fragments from both ends are labeled.

The main idea in the new algorithm is to first identify all *complementary pairs* of primary fragments. The sum of distances of a complementary pair of primary fragments will add up to the length of the parent fragment. There will be $n - 2$ pairs of complementary primary fragments, apart from the the largest fragment (between the two end markers) which doesn't form a pair. Notice that there can be other pairs of fragments which might add up to the length of the parent fragment, but they won't be primary fragments. We can easily identify the $n - 2$ complementary fragment pairs because these are precisely the fragments which are labeled. Henceforth we assume that we have identified the complementary pairs denoted by set $C = \{c_1, \ldots, c_{n-2}\}$. Each complementary pair corresponds to a restriction site. We denote the primary fragments of the complementary pair c_i by (x_i, y_i), with $x_i \leq y_i$. By definition, $x_i + y_i = t$, where t is the length of the parent fragment. The problem, of course, is to decide the placements of these sites relative to one another. That is, if we treat $c_1, \ldots, c_{n-2}$ as boolean variables (we abuse notation to denote c_i to be the complementary pair (x_i, y_i) as well as the boolean value c_i indicating where the site corresponding to the pair occurs), then a value of 0 means that the restriction site corresponding to this complementary pair occurs near the left end, and a value of 1 means that the restriction side occurs near the right end. For example consider the instance shown in figure 1. The following assignment of c values give a valid solution: (3,24): $c = 0$, (11,16): $c = 0$, (10,17): $c = 1$. For example, for the pair (10,17) the c value of 1 implies that that the smaller fragment (10) is at the right end and for (3,24) the value of 0 implies that the smaller fragment (3) is at the left end. Thus an assignment of 0/1 to the c variables leads to a complete reconstruction of the DNA molecule.

First we will give an algorithm for the error-free version of the problem. Here we assume that all the fragment lengths are exactly known and all the $\binom{n}{2}$ fragments are present.

3.1 An Algorithm for Error-free Labeled Partial Digest

The algorithm is given in figure 2 . We will step through the algorithm and give a formal proof of correctness. In step 1, we determine all the complementary pairs $c_i = (x_i, y_i)$; this is easily done since primary fragments are labeled. We process the complementary pairs one by one, by non-increasing order of $\max\{x_i, y_i\}$. The site (point) corresponding to a complementary pair can be in one of two possible locations, towards either the left end or the right end (unless it is exactly in the center, in which case it is trivial). In the ith step of the **for** loop, we determine the placement of the ith complementary pair, $c_i = (x_i, t - x_i)$. We need the following facts and definitions to show how this is done.

By our convention, $x_i \leq t/2$. By the order in which complementary pairs are processed above, all the sites determined so far lie either in the range $[0 \ldots x_i]$ or in the range $[t - x_i \ldots t]$. We will categorize these sites into two types, *symmetric* and *asymmetric*. A site is called *symmetric* if it has a symmetric counterpart on the other side of the center (amongst the sites already placed). In other words, the site corresponding to the pair $(x, t - x)$ is symmetric if there is another complementary pair $(x, t - x)$ which has already been placed. Otherwise the site is *asymmetric*. Note that a site that is asymmetric in the current step could *become* symmetric in the next step when its symmetric counterpart, if any, gets placed. Further, if a site remains asymmetric for two consecutive steps then it remains asymmetric for the entire course of the algorithm (this follows from the order in which complementary pairs are

5

Input: set of all fragment lengths D (including the marked fragments), n
Output: boolean values for $c_i, 1 \leq i \leq n - 2$ (0 means left end, 1 means right end)

1. Identify all the $n - 2$ complementary pairs of primary fragments from D.
Remove them from D. (Now D contains only secondary fragments)

2. Order the complementary pairs $c_i = (x_i, y_i)$ by non-increasing order of $\max\{x, y\}$.

3. Initialize the two endpoints as sites.
(Let t denote the distance between the two endpoints)

4. for $i = 1$ **to** $n - 2$ **do**
    4.1. Let d_i be the distance from the asymmetric site a_i
farthest from the center, to its nearest endpoint.
Let boolean value z_i denote the end which is nearest to a_i.
If no such site exists then d_i is undefined.

    4.2. if d_i is undefined **then**
        $c_i = 0$ (or 1, it does not matter)

    4.3. else
        4.3.1. Let $M = \max\{D\}$, the maximum distance in the set D
        4.3.2. Let $c_i = (x_i, y_i)$, where $x_i \leq t/2$.
        4.3.3. if $(t - x_i - d_i) = M$ **then**
          $c_i = 1 - z_i$
        4.3.4. else if $(t - x_i - d_i) > M$ **then**
          $c_i = z_i$
        4.3.5. else $((t - x_i - d_i) < M)$ **output** "no solution"

    4.4. Discard all the distances from the set D between c_i
and the sites placed till now.

    4.5. if c_i is a symmetric site **then**
(note that c_i is symmetric if its counterpart also gets fixed)
        4.5.1. for $j = i + 1$ **to** $n - 2$ **do**
            4.5.1.1. Let $c_j = (x_j, y_j)$ be a site yet to be placed.
            4.5.1.2. Discard distances $|x_i - x_j|$ and $|x_i - y_j|$ from D.
        (i.e., Discard all the distances from the set D
between c_i and the future site c_j)
        4.5.2. endfor
(If any of the distances are not present in D then no solution exists)

5. endfor

Figure 2: Error-Free Labeled Partial Digest Algorithm

processed).

To place $c_i = (x_i, t - x_i)$, we need to determine the quantity d_i, which is defined to be the distance from the asymmetric site a_i farthest from the center, to its nearest endpoint (i.e., d_i is the minimum of the distance from x_1 and x_n to a_i). This is done in step 4.1. Note that d_i would be undefined if all sites placed so far are symmetric. In this case clearly it does not matter where c_i is placed. We handle this easy case in step 4.3, where we give a value of 0 to c_i (place it on the left end). The nontrivial case is when d_i is defined. Without loss of generality assume a_i is towards the left end. Hence d_i is the distance from the left end to a_i. The following lemma makes the placement of c_i straightforward.

Lemma 3.1 *Let M be defined (in step 4.3.1) as in the algorithm given in figure 2. That is,*
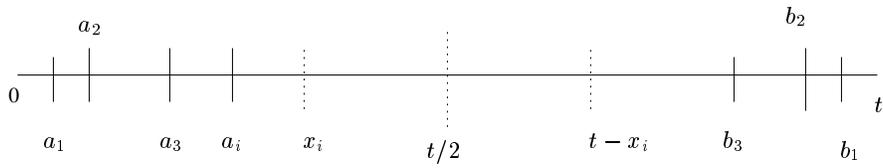
Figure 3: Illustration for proof of lemma 3.1. The complementary pair $(x_i, t - x_i)$ can be placed at two positions shown as dotted lines. a_i is the asymmetric site farthest from the center (point $t/2$). Note that all points $(a_1, a_2, a_3, b_1, b_2, b_3)$ which are farther away from the center are symmetric.

M is the maximum distance in D after removing all distances between sites placed so far and all distances between symmetric sites placed so far and all future sites (including c_i). Then, if $t - x_i - d_i = M$, then c_i must be placed at $t - x_i$, i.e. near the right end. Otherwise if $t - x_i - d_i \geq M$ then it must be placed at x_i, i.e near the left end. Finally if $t - x - d_i < M$, there is no solution.

Proof: See figure 3. M is the largest distance remaining in set D after we have removed all distances between sites placed till the $(i - 1)$th round and all the distances between any symmetric site placed till the $(i - 1)$th round and all future sites to be placed (including c_i). This removal is done in steps 4.4 and 4.5 after we place a site in each step. In step 4.5, the distances can be easily identified and discarded irrespective of where the future sites may be placed due to symmetry. There are three cases possible:

$t - x_i - d_i = M$**:** We claim that c_i must be placed at $t - x_i$. The reason is as follows. Distance M could be realized by sites in one of the following categories:

1. Between two already placed sites.

2. Between two future sites (c_i inclusive).

3. Between a future site (c_i inclusive) and an already placed site.

M cannot be realized by category 1 as these distances have been removed from D (and M still remains). M cannot be realized by category 2 as distances between these sites are all at most $t - 2x_i < t - x_i - d_i$. So M must be realized by category 3. M cannot have a site which is processed *prior* to a_i as its endpoint, because such a site would be a symmetric site (by the definition of a_i) and distances from this site to all future sites have been removed from D. M cannot be realized by an already placed site processed *after* a_i and a future site (c_i inclusive) as these distances are smaller than M. Thus the only way M can be realized is by placing c_i at $t - x_i$.

$t - x_i - d_i > M$**:** We claim that c_i must be placed at x_i (near the left end). This is because placing c_i at $t - x_i - d_i$ contradicts the definition of M being the largest distance in D.

$t - x_i - d_i < M$**:** In this case there can be no solution as the distance M can never be realized since all future points (including c_i) must have a distance smaller than M from a_i.

$\square$

3.2 Running Time

The running time of the error-free labeled partial digest algorithm is easy to analyze. Step 1 takes $O(n^2)$ time as it involves scanning $O(n^2)$ fragments and determining the ones that are labeled. Step 2 can be done in $O(n \log n)$ time by sorting. Steps 4.1, 4.2, 4.3.1 involve operations of searching, deleting and finding a maximum in a set of size $O(n^2)$. Steps 4.4 and 4.5 involve $O(n)$ delete operations per round. Each operation can be implemented in $O(\log n)$ time (by a balanced binary tree) and there are a total of $O(n^2)$ search and delete operations ($O(n)$ per loop), and there are a total of $O(n)$ find-max operations. Hence there are a total of $O(n^2)$ operations and the total time is $O(n^2 \log n)$.

3.3 Unique Reconstruction

We can show that the reconstruction is unique up-to lateral inversion.

Lemma 3.2 *The labeled partial digest algorithm reconstructs the sites of the DNA molecule in exactly the same relative fashion, up-to lateral inversion.*

Proof: We can show this by induction on i, the for loop variable in the algorithm. In the ith step we determine the placement of complementary pair c_i. If c_i has a symmetric counterpart, then we note that the relative placement of c_i does not matter with respect to other sites. On the other hand if c_i is asymmetric then we determine the placement of c_i in step 4.3, where there is only one choice (if a solution exists). $\square$

Thus we can state the following theorem.

Theorem 3.1 *The algorithm in figure 2 solves the (error-free) labeled partial digest problem in time $O(n^2 \log n)$. Further the reconstruction is unique (up to lateral inversion).*

4 Tolerating Errors in Fragment Lengths

We analyze the tolerance of our restriction mapping technique to errors in fragment lengths. As pointed out earlier, a restriction mapping algorithm has to deal with inexact fragment lengths due to gel electrophoresis. In this section, we assume that a valid reconstruction always exists.

We use interval arithmetic in our algorithm to handle inexact lengths [15]. Suppose we assume a given *absolute* error e. Then the inexact length of a fragment lies in an interval $[f - e, f + e]$, where f is the true length.. We would like to calculate a bound on the error which our technique can tolerate, while still giving unique reconstruction.

Let Δ denote the *minimum* inter-site distance, i.e., the minimum distance separating two successive restriction sites. We show that the technique is robust in tolerating an absolute error up to $e = \frac{\Delta}{6n+2}$, i.e., it reconstructs the sites in the "same" relative ordering as in the error-free case.

Assumption. We first give an algorithm for the case when there are no symmetric (or near-symmetric) sites. That is, if $c_i = (x_i, y_i)$ and $c_j = (x_j, y_j)$ ($x_i \leq y_i$ and $x_j \leq y_j$) are any two sites, then $|x_i - x_j| \geq \Delta$. Later, we will discuss how to handle the case when there are symmetric sites.

8

Input: set of all fragment lengths D (including the marked fragments), n
Each fragment length (given by experiment) has an absolute error e less than $\frac{\Delta}{6n+2}$,
where Δ is the minimum inter-site distance.
Also, for any two sites $c_i = (x_i, y_i)$ and $c_j = (x_j, y_j)$, $|x_i - x_j| \geq \Delta$
(i.e., no symmetric or near symmetric sites).
Output: boolean values for $c_i, 1 \leq i \leq n-2$ (0 means left end, 1 means right end)

1. Identify all the $n-2$ complementary pairs of primary fragments from D, using lemma 4.1.
Remove them from D. (Now D contains only secondary fragments.)

2. Order the complementary pairs $c_i = (x_i, y_i)$ by non-increasing order of $\max\{x, y\}$.

3. Initialize the two endpoints as sites.

4. Initialize the interval set $S_0 = \{\}$ and edge set $E_0 = \{\}$.
Initialize the bipartite graph $G_0 = (S_0, D, E_0)$.

5. for $i = 1$ **to** $n-2$ **do**
    5.1. Let d_i be the distance from the asymmetric site a_i
    farthest from the center, to its nearest endpoint.
    Let boolean value z_i denote the end which is nearest to a_i.
    5.2. Let $c_i = (x_i, y_i)$, where $x_i \leq t/2$.
    5.3. Let s be the interval $(y_i - d_i - 3e, y_i - d_i + 3e)$.
    Let F be the set of edges between s and fragments that lie in the interval s.
    (F can be empty.)
    Let G be the bipartite graph $(S_{i-1} \cup \{s\}, D, E_{i-1} \cup F)$.

    5.4. if G is consistent **then**
        $c_i = 1 - z_i$
    5.5. else (c_i should be placed on the other side)
        $c_i = z_i$
    (Update the bipartite graph)

    5.6. $S_i = S_{i-1}$, $E_i = E_{i-1}$
    5.7. for $j = 1$ **to** $i-1$ **do**
        5.7.1. Let $c_j = (x_j, y_j)$ be the jth site already placed.
        5.7.2. if $c_j = c_i$ **then**
        $S_i = S_i \cup \{(|x_j - x_i| - 3e, |x_j - x_i| + 3e)\}$.
        5.7.3. else
        $S_i = S_i \cup \{(|y_j - x_i| - 3e, |y_j - x_i| + 3e)\}$.
    5.8. endfor
    5.9. Update E_i to include all edges between (added) intervals and
    fragments whose lengths lie in these intervals.

6. endfor

Figure 4: Robust Labeled Partial Digest Algorithm

The Algorithm. The robust labeled partial digest algorithm is given in figure 4. We will now explain the crucial steps of the algorithm. We will use the following notation: if length is denoted by a primed variable then it represents the true length, otherwise it is inexact.

The following lemma shows that we can identify complementary primary fragments correctly when the absolute error is less than $\Delta/6$.

Lemma 4.1 *Let f_1 and f_2 be the lengths of two (labeled) primary fragments P_1 and P_2 respectively and let t be length of the parent fragment P (all are inexact fragment lengths*

given by experiment). Then P_1 and P_2 form a complementary pair if and only if $f_1 + f_2$ lies in the (open) interval $(t - \frac{3\Delta}{6}, t + \frac{3\Delta}{6})$.

Proof: Let f_1', f_2' and t' represent the true lengths of P_1, P_2 and P respectively. First, suppose P_1 and P_2 form a complementary pair. Then $f_1' + f_2' = t'$. Since absolute error for a fragment is less than $\Delta/6$, $f_1 + f_2$ lies in the interval $(f_1' + f_2' - \frac{2\Delta}{6}, f_1' + f_2' + \frac{2\Delta}{6})$. Since t lies in the interval $(t' - \frac{\Delta}{6}, t' + \frac{\Delta}{6})$ it follows that $f_1 + f_2$ lies in $(t - \frac{3\Delta}{6}, t + \frac{3\Delta}{6})$. Now we show the converse. Suppose $f_1 + f_2$ lies in the (open) interval $(t - \frac{3\Delta}{6}, t + \frac{3\Delta}{6})$ and suppose P_1 and P_2 do not form a complementary pair, i.e., $f_1' + f_2' \neq t'$. Hence $|t' - (f_1' + f_2')| \geq \Delta$ (by virtue of the minimum inter-site distance and by virtue of the above assumption; note that both facts are required). Since the absolute error is less than $\Delta/6$ we have $|t - (f_1 + f_2)| \geq \frac{3\Delta}{6}$. Hence $f_1 + f_2$ cannot lie in $(t - \frac{3\Delta}{6}, t + \frac{3\Delta}{6})$, a contradiction to our assumption. $\square$

It can be easily shown that, under an absolute error less than $\Delta/6$, the complementary pairs are sorted in the same order (in step 2) as in the error-free case. As before, Step 5 processes complementary pairs in this order, determining the placing of the ith pair in the ith round. In the error-free algorithm, recall that this determination required identifying and removing from consideration secondary fragments between already placed sites. This computation is non-trivial now due to the presence of errors; in particular, the secondary fragment corresponding to a particular pair of already placed sites cannot be identified uniquely any longer to due the presence of fragments with almost equal lengths. To tackle this issue, we set up the following framework.

The Secondary Fragment Matching Graph. Consider two complementary pairs $c_i = (x_i, y_i)$ and $c_j = (x_j, y_j)$. There is a unique secondary fragment whose endpoints are these two sites. We say that this secondary fragment is *induced* by these two sites. If c_i and c_j are placed on opposite sides then this secondary fragment has length $|y_j' - x_i'|$ and if c_i and c_j are placed on the same side, then this secondary fragment has length $|x_j' - x_i'|$. If the absolute error is bounded by e, then it is easily seen that the erroneous length of this secondary fragment lies in the interval $(l - 3e, l + 3e)$, where $l = |y_j - x_i|$ or $l = |x_j - x_i|$, depending on which of the two cases above occurs.

Suppose, we would like to remove from consideration the secondary fragment induced by already placed sites c_i, c_j. This fragment could be any of the secondary fragments having length in the interval $(l - 3e, l + 3e)$. If there is more than one fragment (in D) whose length lies in the above interval, we have to decide which one to "assign" to this interval. Committing any one of them might cause problems in placing future sites. In particular, errors can add up. To avoid this, we don't commit a fragment to one particular interval, but maintain a set of fragments corresponding to different intervals. This is done by constructing a bipartite graph for a given set of intervals.

Let S be a given set of intervals, each interval corresponding to a pair of sites. Let T be the set of all secondary fragments. Then the bipartite graph $G = (S, T, E)$ with edge set E, is formed by connecting a fragment in T to an interval in S if the fragment lies in that interval.

The bipartite graph is used in implicitly assigning fragments to intervals. We call G to be *consistent* if there is a complete matching of S on to T, i.e, all vertices in S must be matched to a unique vertex in T. Clearly, if G is consistent then each interval in the set S has a unique fragment lying in it. It is also easy to see that if a valid reconstruction exists

then the bipartite graph for the set S consisting of all the $\binom{n-2}{2}$ pairs of sites, should be consistent.

Before we prove our main lemma, we need an observation on overlapping intervals. Let s_i (for any index i) be an interval corresponding to a pair of sites as described above. Two intervals overlap if the corresponding intervals on the real line overlap.

Observation 4.1 *Let $s_1, s_2, s_3, \ldots$ be a sequence of successive overlapping intervals, i.e., s_1 overlaps with s_2, s_2 with s_3, and so on. If the absolute error e is less than $\frac{\Delta}{6n+2}$, then the number of intervals in the overlapping sequence can be at most $n-1$.*

Proof: Suppose the number of intervals in the above sequence is n (or larger, in which case consider only the first n intervals in the sequence). We will derive a contradiction, as follows.

Consider any two intervals in the above sequence, each interval corresponding to a pair of sites. Consider the secondary fragments induced by each pair of sites. We claim that these two secondary fragments cannot be *nested*, i.e., one secondary fragment cannot appear completely inside the other. It immediately follows that the number of intervals in the above sequence is at most $n-1$, a contradiction.

Suppose the above two secondary fragments were nested. Then their real lengths would differ by at least Δ. The difference in their erroneous lengths would be at least $\Delta - 2e$. But since the erroneous lengths lie in two intervals in the above sequence of n successive overlapping intervals, the difference in these erroneous lengths is less than $6en$, which is less than $\Delta - 2e$, for $e < \frac{\Delta}{6n+2}$. The claim follows. $\square$

The next lemma (analogous to lemma 3.1) shows the correctness of placement of c_i in steps 5.4 and 5.5. It also follows from the lemma that c_i is placed in the same relative position as in the error-free case.

Lemma 4.2 *Let the absolute error be less than $e = \frac{\Delta}{6n+2}$. Assume we are processing the (ith) complementary pair $c_i = (x_i, y_i)$ in step 5 of the algorithm of figure 4. Let d_i be defined as in step 5.1. Let the bipartite graph G be defined as in step 5.3. Then c_i must be placed near the right end if G is consistent, otherwise it must be placed near the left end.*

Proof: We assume by induction that the first $i-1$ sites have been correctly placed. The base case is easily shown to be true. We now show that the ith site is correctly placed by the decision in step 5.4.

As before, we assume without loss of generality that a_i, the farthest asymmetric site from the center, is placed to the left. Let us tentatively place c_i to the right. The graph G in consideration currently has a vertex for each interval corresponding to a pair of already placed sites (i.e., the first $i-1$ sites along with c_i placed tentatively to the right). If G is inconsistent then clearly the above tentative choice of placing c_i on the right is not valid. Therefore, since we are assuming that a valid reconstruction exists, c_i is placed to the left.

Next, suppose G is indeed consistent. Then there must be a fragment of length, say M, uniquely assigned to the interval $(y_i - d_i - 3e, y_i - d_i + 3e)$. We claim that c_i must be placed at the right end. Suppose this is not true, i.e., c_i's correct placement is to the left. Then we will construct a sequence of successive overlapping intervals with the following properties:

1. The first interval in this sequence is $(y_i - d_i - 3e, y_i - d_i + 3e)$.

11

2. All intervals in the sequence correspond to pairs of already placed sites, so they are represented by vertices in G and have unique fragments assigned to them.

3. The fragment M_l corresponding to the last interval in this sequence must be realized by a future site (i.e., one of the endpoints of this fragment must be either the site c_i placed to the left or one of the future sites to be placed).

From observation 4.1, it follows that M_l must have length at least

$$y_i - d_i + 3e - 6(n-1)e > y_i - d_i + 3e - 6(n-1)\frac{\Delta}{6n+2} > y_i - d_i + 3e - \Delta$$

We show a contradiction to this property using the fact that M_l is realized by a future site. Recall our assumption that the inter-site distance is at least Δ and there are no near-symmetric sites. It follows that any fragment realized by a future site (in particular M_l) must have length at most $y_i - d_i - \Delta + 3e$, a contradiction.

It remains to show how to construct the above sequence of intervals when G is consistent, given that the correct placement for c_i is to the left. This is done as follows. The fragment M must be realized by some pair of sites in the correct solution. If this pair of sites does not have a vertex corresponding to it in G, then M is realized by a pair of sites one of which is a future site (i.e., one of the endpoints of this fragment must be either the site c_i placed to the left or one of the future sites to be placed); our sequence construction stops here in this case. Otherwise, if this pair of sites has a vertex corresponding to it in G, then the corresponding interval is added to the above sequence. Next, let M_1 be the fragment assigned to this vertex. M_1 itself must be realized by some other pair of sites; we repeat the above argument with this pair of sites. Note that successive intervals in the resulting sequence share a fragment and are therefore overlapping. $\square$

The running time of the robust algorithm is dominated by step 5.4 in which we check for consistency by looking for a complete bipartite matching in G. We note that in each iteration of the **for** loop, we need to find whether there exists an augmenting path which includes the new interval added (s, in step 4.3). In iteration i, this can take at most $O(i^2n)$ time, since there are at most $O(i^2n)$ edges in G (because by observation 4.1, each interval can have at most $n-1$ fragment lengths lying in it and there are $O(i^2)$ intervals). Thus the overall running time of the robust algorithm is $O(n^4)$.
Thus we state the following theorem.

Theorem 4.1 *We are given an instance of the labeled partial digest problem: $\binom{n}{2}$ fragments with each fragment having an absolute error less than $\frac{\Delta}{6n+2}$, where Δ is the minimum inter-site distance. Further there are no symmetric or near symmetric sites, i.e., for two sites $c_i = (x_i, y_i)$ and $c_j = (x_j, y_j)$, we have $|x_i - x_j| \geq \Delta$. Then the robust labeled partial digest algorithm of figure 4 can reconstruct the sites uniquely in $O(n^4)$ time.*

Remarks on the Assumption. We now remark on the case where there are near-symmetric sites in the DNA. We note that we need only one "well-separated" asymmetric site (site $a_i = c_1$ in our analysis) for our results to hold. Thus we need only to assume that c_1 is an asymmetric site, i.e., $|x_1 - x_2| \geq \Delta$. If there are very near symmetric sites (i.e., when $|x_1 - x_2| \leq 2e$), it can be seen that our algorithm will continue to work correctly (but for near symmetric sites possibly getting exchanged), as long as the site c_1 is asymmetric. We

can easily check whether c_1 is asymmetric by inspecting the complementary pairs of c_1 and c_2. How can one ensure that site c_1 is asymmetric? An easy way is to "add" a fragment of length at least Δ to the parent fragment and then do the digest. Of course, we have to take care that the added fragment itself does not have any restriction sites. Typically, the target DNA (used for restriction analysis) itself is part of a much larger DNA molecule. Thus in practice, instead of "adding" fragments, the target DNA itself can be suitably cut to ensure that c_1 is asymmetric.

5   Experimental Results

Our error analysis in the previous section was a worst-case analysis. To see how the algorithm might perform in practice we decided to test our algorithm on a real DNA molecule: the bacteriophage λ [12], a common cloning vehicle with seven restriction sites for the enzyme *Hind*III. The distance between adjacent sites on this molecule are

$$\{23130, 2027, 2322, 9416, 564, 125, 6557, 4361\}$$

which leads to 36 following pairwise distances:
125, 564, 689, 2027, 2322, 4349, 4361, 6557, 6682, 7246, 9416, 9980, 10105, 10918, 11043, 11607, 11738, 12302, 12427, 13765, 14329, 14454, 16662, 18984, 21011, 21023, 23130, 23345, 25157, 25372, 27479, 36895, 37459, 37584, 44141, 48502.

The complementary pairs with their respective c values are shown in figure 5.

complementary pair	c value
(4271, 44231)	0
(10828, 37674)	0
(11043, 37549)	0
(11607, 36895)	0
(21023, 27479)	0
(23345, 25157)	0
(23130, 25372)	1

Figure 5: Restriction sites for the enzyme *Hind*III on the bacteriophage λ listed as complementary pairs with their respective c values.

As in [15] we perturbed the length of each of these fragments to simulate the effects of experimental error on the performance of our algorithm. Specifically to simulate a relative error of r, each fragment of length f was replaced with a random integer selected uniformly from the interval $[f.(1-r), f.(1+r)]$. We used values of r ranging from 0% to 5%. Currently, 2% to 5% relative error is achievable in the laboratory [12]. The results using the algorithm of figure 2 are given in figure 6. For this experiment we assume that all 36 fragment lengths are given. A trial is a success if the reconstructed sites agreed with the true (relative) location of the sites. For each value of r success percentage was computed by doing 50 trials. The reconstruction time for all the trials was less than a second on a SUN Ultra 10 machine.

In [15] the idea of *grouping* fragments of almost similar lengths is used to reduce redundant computation at the cost of potentially missed solutions. The performance of their method

rapidly deteriorates with decrease in grouping percentage. In our experiments, if more than one fragment is present for a given interval x we choose the "best" fragment for that interval by using the rule : choose the fragment of length l such that $|l - x|$ is minimized.

r value	success percentage
0.0%	100%
1.0%	98%
2.0%	96%
3.0%	94%
4.0%	91%
5.0%	87%

Figure 6: Simulation results on the bacteriophage λ for different relative errors in fragment lengths.

6   The De Novo Peptide Sequencing Problem

Motivated by our new approach to restriction mapping, we present a simple and efficient algorithm for the *de novo peptide sequencing problem* which arises in the problem of reconstructing the amino acid (or the peptide) sequence of a protein [4, 3]. There are many similarities between the peptide sequencing problem and the labeled partial digest problem which will become clear shortly.

The de novo peptide sequencing problem is the reconstruction of the peptide sequence from a given tandem mass spectral data. We briefly describe the methodology and mention recent algorithms for the problem. More details can be found in [3].

To determine the amino acid (peptide) sequence *tandem mass spectrometry* along with microcolumn liquid chromatography has been widely used as follows. A large number of molecules of the same but unknown peptide sequence are separated from one or multiple liquid chromatographers and a mass analyzer. Then they are fragmented and ionized by collision-induced dissociation and all the resulting ions are measured by the mass spectrometer for mass/charge ratios. In the process of dissociation, a peptide bond at a random position is broken, and each molecule is fragmented into two *complementary* ions. The two ions (called as *prefix* and *suffix* henceforth) are complementary because joining them determines the original peptide sequence. The dissociation process yields all possible prefix and suffix subsequences of the original peptide sequence. For example, the disassociation of a peptide $(R1 - R2 - R3)$ leads to the following prefix ions (also called as *b-ions*): $(R1)^+$, $(R1 - R2)^+$ and $(R1 - R2 - R3)^+$ and to the following suffix ions (also called as *y-ions*): $(R1 - R2 - R3)^+$, $(R2 - R3)^+$ and $(R3)^+$. These ions display a spectrum from which their masses can be calculated. All the prefix (or suffix) subsequences form a sequence ladder where the mass difference between two adjacent b-ions (or y-ions) equals the mass of that amino acid. However, the interpretation of the mass spectroscopy data has to take into account the following two factors:

1. it is unknown whether a mass peak of some ion corresponds to a prefix or suffix sequence;

2. some ions may be lost in experiments and the corresponding mass peaks disappear in the spectrum.

Thus the de novo peptide sequencing problem is given an input of a subset of prefix and suffix masses of a target peptide sequence P (whose mass is known) we want a peptide sequence Q such that a subset of its prefixes and suffixes gives the same input masses.

As outlined in [3], an equivalent way of approaching the problem is as follows. We are given the mass W of a target peptide sequence P, k ions $I_1, \ldots, I_k$ of P and the masses $w_1, \ldots, w_k$ of these ions. For each ion I_j it is unknown whether it is a suffix or a prefix ion. It is useful to map ions to coordinates in a real line as follows. We create two sites (or coordinates), 0 and $W - 18$, corresponding to zero mass and the total mass of all amino acids of P respectively. Each ion I_j can correspond to potentially two coordinates $w_j - 1$ or $W - w_j + 1$ depending on whether it is a prefix or a suffix ion. The sequencing problem is to simply place each of the ions in one of it's potential coordinates such that the distance between any two sites adds up to mass of *some* amino acids. In [3], a graph $G = (V, E)$ is created to solve the problem. The vertex set V is the set of all possible coordinates, including the two end sites and an edge exists between sites if the distance between them adds up to the mass of some amino acids. Thus, $|V| = 2k + 2$ and $|E|$ can potentially be $O(|V|^2)$. The paper gives first an $O(|V|^2)$ algorithm to find a feasible solution given G, using dynamic programming. Then they give a more sophisticated $O(|V| + |E|)$ algorithm. To construct G, a mass array $\mathcal{A}$ is used. Given a mass m, $0 < m \leq h$ (where h is the maximum mass under construction, i.e., $m \leq W - 18$), $\mathcal{A}[m] = 1$ if and only if m equals the sum of some amino acid masses. $\mathcal{A}$ can be precomputed in time $O(h/\delta)$ time (Theorem 2 in [3]) where δ is the mass precision. Given A, G can be constructed in $O(k^2)$ time. Thus the best algorithm of [3] takes $O(k^2)$ time to find a solution to the sequencing problem.

The algorithm and its analysis. We now outline an algorithm for the problem, which is very similar to the one we gave for labeled partial digest. Our algorithm is simple and straightforward and takes $O(k \log k)$ time, where k is the number of sites (i.e. mass peaks). As in [3] we precompute the mass array $\mathcal{A}$ and use it to determine feasibility between sites, but we don't construct G. Before we describe our algorithm it is instructive to point out its similarity with the error-free labeled partial digest algorithm (figure 2). We also have complementary pairs (x_i, y_i) representing a site, where $x_i = w_i - 1$ and $y_i = W - w_i + 1$ are the prefix and suffix masses respectively. However, in peptide sequencing, , the feasibility of distances between sites are determined by the mass array $\mathcal{A}$ and not the presence of intermediate fragments as in partial digest.

The algorithm is given in figure 7. We seek an assignment of values to the boolean variables c_i which indicates whether the mass peak corresponds to a prefix ($c_i = 0$) or a suffix ion ($c_i = 1$). For ease of notation, we assume that we are given a function f which takes as input two sites x and y and an orientation r (can be either "same side" or "opposite sides") and returns 1 if the distance between the two sites is feasible in that orientation, and zero otherwise. f simply uses the mass array $\mathcal{A}$ to determine feasibility. We also use the term "clash" to refer to infeasibility of placing two sites in a particular orientation.

We will now comment on a few crucial steps of the algorithm. Step 4 processes all the sites except the last one. Assume that we are processing site i. Let $c_{i-1}, c_{i-2}, \ldots, c_{i-j}$ be the most recent maximal consecutive stretch of sites placed on the same side, and without loss of generality we assume that it is on the right. Let $c_{i-j-1}, c_{i-j-2}, \ldots, c_{i-j-k}$ be the most recent maximal consecutive stretch of sites placed on the left. That is, $c_{i-1} = c_{i-2} = \cdots = c_{i-j} = 1$

and $c_{i-j-1} = \cdots = c_{i-j-k} = 0$ and $c_{i-j-k-1} = 1$ if $i - j - k - 1 > 0$. We always maintain the *invariant* that c_{i-j} and c_{i-j-1} clash when placed on the same side i.e., the first site of the stretch on one side clashes with the last site of the stretch on the other side.

To place site i we first check to see whether it clashes with c_{i-1} when placed on the right, i.e. when they are both on the same right side. If it does not, then we place it on the right. (Note that, if c_i does not clash with c_{i-1} then it does not clash with any other site placed on the right.) This extends the right stretch to include c_i. Otherwise, we place it on the left if it does not clash with c_{i-j-1}. Note that this begins a new stretch on the left. The invariant is also maintained, i.e., c_i, the start of the left stretch clashes with c_{i-1} the most recent site of the right stretch. On the other hand, if c_i clashes with c_{i-j-1} also, then we place c_i on the right (i.e. $c_i = 1$) and *move* some of the sites on the right to the left. That is, we move the sites $c_{i-1} \ldots, c_{i-j'}$, where j' is the maximum integer such that all of the above sites clash with c_i on the right. The invariant again holds. We claim that if $j' = j$, there is no solution. The reason is, c_i clashes with both c_{i-j-1} and c_{i-j} which in turn mutually clash (because of the invariant).

To place the last site c_n, we do the following. As before, let $c_{n-1}, \ldots, c_{n-j}$ be the maximal stretch of sites on the right (w.l.o.g) and $c_{n-j-1}, \ldots, c_{n-j-k}$ be the stretch on the left. We first check if c_n clashes with c_{n-j-1} when placed on the left. We will first assume that it doesn't. Then, if c_n does not clash with c_{n-1}, we are done (step 4.1). But if it clashes with c_{n-1}, we move $c_{n-1}, \ldots, c_{n-j'}$ to the left, where j' is the largest number that all of the above sites (placed on the right) clash with c_n placed on the left. Now, if c_{n-j-1} and $c_{n-j'}$ don't clash (both of them placed on the left) then we are done. Otherwise, we claim that placing c_n on the left is infeasible given c_{n-j-1} is on the left. This is because, c_{n-j-1} and $c_{n-j'}$ cannot be on the same side, $c_{n-j'}$ and c_n must be on different sides and therefore, c_n and c_{n-j-1} must be on different sides. Thus, if we don't succeed placing c_n on the left, it means that c_n and c_{n-j-1} cannot be on the same side. So we now try placing them on opposite sides with c_n on the right. If c_n clashes with c_{n-j-1} then clearly there is no solution. Suppose it doesn't. We now check whether c_n clashes with c_{n-1}. If it doesn't then we are done (step 5.1). Otherwise, we move $c_{n-1}, \ldots, c_{n-j'}$ to the left, where j' is the largest number such that all the above sites clash with c_n placed on the right. We claim that $j' < j$, otherwise, c_n and c_{n-j} cannot be on the same side, c_{n-j} and c_{n-j-1} cannot be on the same side by the invariant and c_n and c_{n-j-1} also cannot be on the same side, by our argument above. Now, if $c_{n-j'}$ does not clash with c_{n-j-1} then we are done (step 5.2), else there is no solution. This is because c_n and $c_{n-j'}$ cannot be on the same side, $c_{n-j'}$ and c_{n-j-1} cannot be on the same side and thus c_n and c_{n-j-1} have to be on the same side, which is a contradiction.

Running time. We assume, as mentioned earlier, that determining feasibility between a pair of sites takes $O(1)$ time since we have precomputed the mass array $\mathcal{A}$. Step 1 takes $O(k \log k)$ time [2] due to sorting. Steps 3 and 4 can be implemented in $O(k)$ time, by noting that each site is handled at most twice (once when it is placed, and probably once more if it is to be moved to the other side). Thus, maintaining a couple of pointers which move inward starting from either ends, the total time taken for steps 3 and 4 is $O(k)$.

---

[2]Actually, since h/δ is comparable to k, we can use bucket sort to sort the complementary pairs. Thus the time taken to sort can be subsumed in the time taken to compute the mass array $\mathcal{A}$. However, for clarity, we exclude the time taken to construct $\mathcal{A}$ from the sequencing algorithm.

Input: set of $n - 2$ sites (i.e. each mass peak as a complementary pair (x_i, y_i)) and the function f.

Output: boolean values for $c_i, 3 \leq i \leq n$ (0 means prefix i.e., left end , 1 means suffix, i.e., right end).

1 Order the sites $c_i = (x_i, y_i)$ by non-increasing order of $\max\{x, y\}$.

2 Initialize the two endpoints as sites, i.e., $c_1 = 0$ and $c_2 = 1$

3 for $i = 3$ **to** $n - 1$ **do**

    3.1 Let $c_{i-1}, c_{i-2}, \ldots, c_{i-j}$ be the most recent maximal consecutive set of sites placed on the right (without loss of generality) and let $c_{i-j-1}, c_{i-j-2}, \ldots, c_{i-j-k}$ be the maximal consecutive set of sites placed on the left, where $1 \leq j < i, k \geq 1$.

    3.2 if $f(c_i, c_{i-1}, \text{“}sameside\text{''}) = 1$
(i.e., the distance between c_i and c_{i-1} is feasible when c_i is placed on the right)

        $c_i = 1$ (place it on the right)

    3.3 else if $f(c_i, c_{i-j-1}, \text{“}sameside\text{''}) = 1$

        $c_i = 0$ (place it on the left)

    3.4 else

        $c_i = 1$
Let $j' < j$ be the largest index such that all of $c_{i-1}, \ldots, c_{i-j'}$ clash with c_i on the right
Move $c_{i-1}, \ldots, c_{i-j'}$ to the left (i.e., change these c values to 0)
(if $j' = j$, there is no solution)

endfor

(to place the last site c_n)

Let $c_{n-1}, c_{n-2}, \ldots, c_{n-j}$ be the most recent maximal consecutive set of sites placed on the right (w.l.o.g) and let $c_{n-j-1}, c_{n-j-2}, \ldots, c_{n-j-k}$ be the maximal consecutive set of sites placed on the left, where $1 \leq j < n, k \geq 1$.

4 if $f(c_n, c_{n-j-1}, \text{“}sameside\text{''}) = 1$ **then**

    4.1 If $f(c_n, c_{n-1}, \text{“}oppositeside\text{''}) = 1$ **then**

        $c_n = 0$ (c_n does not clash with c_{n-1} as well as c_{n-j-1})

    4.2 else

        $c_n = 0$
Move $c_{n-1}, \ldots, c_{n-j'}$ where j' is the largest index such that all of $c_{n-1}, \ldots, c_{n-j'}$ clash with c_n placed on the left
(if $f(c_{n-j'}, c_{n-j-1}, \text{“}sameside\text{''}) = 0$ then c_n cannot be 0)

5 else

    $c_n = 1$
(if c_n clashes with c_{n-j-1} then there is no solution)

    5.1 if $f(c_n, c_{n-1}, \text{“}sameside\text{''}) = 1$ **then**

        $c_n = 1$

    5.2 else

        Move $c_{n-1}, \ldots, c_{n-j'}$ to the left, where j' is the largest index such that all of $c_{n-1}, \ldots, c_{n-j'}$ clash with c_n on the right
(if $j' = j$ or $f(c_{n-j'}, c_{n-j-1}, \text{“}sameside\text{''}) = 0$ then there is no solution)

Figure 7: Algorithm for peptide sequencing

7 Conclusion

We have presented a new technique for finding restriction sites in a DNA molecule. The technique uses partial digest and can be easily implemented in the laboratory. We showed

that this reconstruction technique has many advantages including fast ($O(n^2 \log n)$ worst case time) algorithm, besides giving a unique reconstruction. It is also robust in dealing with experimental errors in fragment lengths. We showed that it can provably tolerate an absolute error of $O(\frac{\Delta}{n})$ while still giving a unique reconstruction.

Another important contribution of the paper is a simple and efficient algorithm for the de novo peptide sequencing problem. Our algorithm, motivated by the error-free labeled partial digest algorithm, improves over the previous algorithm of Chen et. al [3] which uses a more sophisticated dynamic programming based approach.

We conclude with two important open problems in the partial digest approach.

1. Suppose there are multiple copies of each fragment, i.e., different numbers of copies for different fragment, then our algorithm as stated above does not work. (This also includes the possibility of distances being lost – missing fragments). This is an important error which can complicate the reconstruction process. This confusion usually arises because fragments of approximately equal length are unlikely to be discriminated on the gel. Hence, there arises uncertainty in the multiplicities of distances.

2. We also would like to point out that the status of the partial digest problem is still unresolved. It is strongly suspected that there is a polynomial time algorithm [14]. We have shown that there is one if we label both ends. It will be interesting if this idea can be somehow extended to solve the unlabeled problem. The difficulty lies in the possible presence of "spurious" complementary pairs, i.e pairs of secondary fragments whose lengths add up to the parent fragment length.

Acknowledgments

References

[1] L. Allison and C.N. Yee. Restriction site mapping is in separation theory. *Comput. Appl. Biol. Sci.*, **4**, 97-101, 1988.

[2] J. Blazewicz, P. Formanowicz, M. Kasprzak, M. Jaroszewski and W.T. Markiewicz. Simplified Partial Digest Method for DNA Restriction Maps Construction. *Bioinformatics*, **17**(5), 398-404, 2001. (Also appeared as a poster in *The Fifth International Conference on Computational Biology (RECOMB)*, 2001.)

[3] T. Chen, M. Kao, M. Tepel, J. Rush and G.M. Church. A Dynamic Programming Approach to De Novo Peptide Sequencing via Tandem Mass Spectrometry. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 389-398, 2000.

[4] V. Dancik, T.A. Addona, K.R. Clauser, J.E. Vath, and P.A. Pevzner. De Novo Peptide Sequencing via Tandem Mass Spectrometry: A Graph-Theoretical Approach. In *Proceedings of the 3rd Annual International Conference on Computational Molecular Biology (RECOMB)*, 1999.

[5] T. Dakic. On the Turnpike Problem. *Ph.D Thesis*, Simon Fraser University, August 2000.

[6] T.I. Dix and D.H. Kieronska. Errors between sites in restriction site mapping. *Comput. Appl. Biol. Sci.*, **4**, 117-123, 1988.

[7] L. Goldstein and M.S. Waterman. Mapping DNA by Stochastic Relaxation. *Advances in Applied Mathematics*, **8**, 194-207, 1987.

[8] R.M. Karp and R. Shamir. Algorithms for Optical Mapping. In *Proceedings of the 2nd International Conference on Computational Biology (RECOMB)*, 117-124, 1998.

[9] L. Newberg and D. Naor. A Lower Bound on the Number of Solutions to the Probed Partial Digest Problem. *Advances in Applied Mathematics*, **14**, 172-183, 1993.

[10] P.A. Pevzner. *Computational Molecular Biology: An Algorithmic Approach*, MIT Press, Cambridge, 2000.

[11] J. Rosenblatt and P. Seymour. The structure of homometric sets. *SIAM J. Alg. Disc. Methods*, **3**, 343-350, 1982.

[12] J. Sambrook, E.F. Fritsch and T. Maniatis. *Molecular Cloning, a laboratory manual*, Second Edition, Cold Spring Harbor Laboratory Press, NY, 1989.

[13] D.C. Schwartz, X. Li, L.I. Hernandez, S.P. Ramnarain, E.J. Huff and Y.K. Wang. Ordered Restriction Maps of Saccharomyces Cerevisiae Chromosomes Constructed by Optical Mapping. *Science*, **262**, 110-114, 1993.

[14] S.S. Skiena, W.D. Smith, and P. Lemke. Reconstructing Sets from Interpoint Distances. In *Proceedings of the 6th ACM Symposium on Computational Geometry*, 332-339, 1990.

[15] S.S. Skiena and G. Sundaram. A Partial Digest Approach to Restriction Site Mapping. *Bulletin of Mathematical Biology*, **56(2)**, 275-294, 1994.

[16] M. Stefik. Inferring DNA structures from segmentation data. *Artificial Intelligence*, **11**, 85-114, 1978.

[17] M.S. Waterman. *Introduction to computational biology : maps, sequences and genomes*, First Edition, Chapman and Hall, NY, 1995.

[18] M.A. Weiss. *Data Structures and Algorithm Analysis in C++*, Benjamin/Cummings, 1994.

[19] Z. Zhang. An Exponential Example for a Partial Digest Mapping Algorithm, *Journal of Computational Biology*, **1(3)**, 235-239, 1994.