The Simplified Partial Digest Problem: Enumerative and Dynamic Programming Algorithms

Jacek Blazewicz, Edmund K. Burke, Marta Kasprzak, Alexandr Kovalev, and Mikhail Y. Kovalyov

Abstract—We study the Simplified Partial Digest Problem (SPDP), which is a mathematical model for a new simplified partial digest method of genome mapping. This method is easy for laboratory implementation and robust with respect to the experimental errors. SPDP is NP-hard in the strong sense. We present an $O(n2^n)$ time enumerative algorithm (ENUM) and an $O(n^{2q})$ time dynamic programming algorithm for the error-free SPDP, where n is the number of restriction sites and q is the number of distinct intersite distances. We also give examples of the problem in which there are $2^{\frac{n+2}{3}-1}$ noncongruent solutions. These examples partially answer a question recently posed in the literature about the number of solutions of SPDP. We adapt our ENUM for handling SPDP with imprecise input data. Finally, we describe and discuss the results of the computer experiments with our algorithms.

Index Terms—Algorithm design and analysis, dynamic programming, genome mapping, restriction site analysis, imprecise information.

1 Introduction

A genome of a living organism can be viewed as a DNA molecule in the form of a *double helix* consisting of two strands. This molecule is a chain of nucleotides called adenine (A), cytosine (C), guanine (G), and thymine (T). Mathematically, its linear structure can be represented as a word in the alphabet $\{A, C, G, T\}$. According to the fundamental law of DNA construction discovered by Watson and Crick [28], one strand of a DNA molecule unambiguously determines its second strand.

Linear structure is an important characteristic of a DNA molecule. At present, it cannot be determined directly by using physical or chemical measurement methods. The existing indirect methods usually include three main hierarchical procedures: mapping, assembling, and sequencing. In a mapping procedure, a DNA molecule is exposed to specific chemicals called restriction enzymes (ferments). Enzymes cut DNA molecules at particular patterns of nucleotides called restriction sites. For example, enzyme *Eco*RI cuts at the pattern $GAATTC$ (see Skiena and Sundaram [26]). During the cutting process, the information about the location of the restriction sites is lost. The available information is the multiset of lengths of the cut fragments (counted in the number of nucleotides between the corresponding restriction sites). These lengths are obtained by a gel electrophoresis experiment that is based on the fact that shorter fragments generate a longer distance in the gel under the electric current.

Reconstructing the location of restriction sites is the subject of a mathematical theory called *restriction site analysis*; see Setubal and Meidanis [24], Waterman [27], or Pevzner [22] for details. The input data for restriction site analysis are the lengths of the cut fragments and appropriate information about the cutting (digesting) method. The most common cutting methods are *double digest*, where two restriction enzymes are used (see, for example, [27] or [22]), and *partial digest*, where the DNA is cut by one enzyme but with different reaction times. The inventor of the partial digest approach was Daniel Nathans, together with his coworkers (see Danna and Nathans [8] and Danna et al. [9]), who, in 1978, received the Nobel Prize for his work on restriction enzymes and restriction mapping. In the following, we will be concerned with this last approach as the double digest constructs too many equivalent solutions (maps); see Waterman [27].

In the classical partial digest approach, a series of digesting experiments is performed. In the first experiment, identical copies of a DNA chain, called DNA clones, are exposed to an enzyme for a sufficiently small time period to cut them in at most one restriction site. The second reaction has a little more time allowed in order to obtain two cuts per clone. Every other experiment takes more time and, finally, the last one takes a sufficiently long time to cut clones in every appropriate site. The *Partial Digest Problem (PDP)* is a mathematical model that aims to reconstruct the map of the target DNA based on the DNA fragment lengths between every two restriction sites. It was studied in the ideal *error-free* case and in the case of experimental errors; see Skiena et al. [25], Skiena and Sundaram [26], Cieliebak et al. [7], and Cieliebak and Eidenbenz [6]. From the

- *J. Blazewicz, M. Kasprzak, and A. Kovalev are with the Institue of Computing Science, Poznan University of Technology, Piotrowo 2, 60-965 Poznan, Poland. E-mail: {jblazewicz, marta, akovalev}@cs.put.poznan.pl.*
- *E.K. Burke is with the School of Computer Science, University of Nottingham, Jubilee Campus, Nottingham NG8 2BB, UK. E-mail: ekb@cs.nott.ac.uk.*
- *M.Y. Kovalyov is with the Belarusian State University and the United Institute of Informatics Problems, Nozavisimosti 4, 220030 Minsk, Belarus. E-mail: koval@newman.bas-net.by.*

Manuscript received 24 Feb. 2006; revised 18 Aug. 2006; accepted 13 Dec. 2006; published online 13 Feb. 2007.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-0026-0206.
Digital Object Identifier no. 10.1109/TCBB.2007.1060.

combinatorial point of view, the computational complexity of PDP is yet to be established. This problem was proved to be NP-hard in the cases of measurement errors [6] and noisy data [7]. However, a proof of NP-hardness of the original error-free PDP, as well as a polynomial-time solution algorithm for it, is not known; see Daurat et al. [10] and Gerard [15].

The main disadvantage of the partial digest approach is that the experimental data is very hard to obtain. The output of the partial digestion is to be a multiset of all *interpoint distances*, where the points are the restriction sites and the ends of the molecule. For n restriction sites, this multiset must consist of $\binom{n+2}{2} = \frac{(n+1)(n+2)}{2}$ fragment lengths. Pevzner [22] writes that the partial digestion has never been the favorite mapping method in biological laboratories because of difficulty in obtaining fragments between every pair of sites. This is confirmed by the statistical data—experiments using this approach are performed on a rather small scale for molecules containing less than 20 restriction sites; see Dudez et al. [13], Keis et al. [18], and Kuwahara et al. [19].

A *simplified partial digest method* was recently proposed by Blazewicz et al. [2] to overcome the disadvantage of the partial digest approach. In this simplified method, one enzyme is used on two sets of clones of the same DNA molecule. The corresponding experiment consists of two parts. In the first part, the time of the chemical reaction is chosen so that target cloned molecules of the first set are cut at one restriction site at most. In the second part, the reaction time span is sufficiently long to cut the cloned molecules of the second set at all restriction sites. The beneficial effect of this simplified approach is not only the reduction of the number of reactions performed but also a much easier choice of the reaction times—they are either very short or very long and there is nothing in between. Experimental data provided by any digesting method can contain a level of error that is proportional to the total amount of data produced. From this point of view, the simplified partial digest method is again beneficial because it produces less experimental data.

This paper presents new combinatorial algorithms that can be used in the restriction site analysis based on the simplified partial digest experiment. The extensive set of experiments verifies the high efficiency of the proposed algorithms and their clear advantage over the existing procedures for realistic practically justified data.

The organization of the paper can be outlined as follows: A mathematical model for genome mapping based on the simplified partial digest experiment is described in Section 2. The model is called the *Simplified Partial Digest Problem (SPDP)*. Section 3 presents an enumerative algorithm (ENUM) for SPDP and examples of this problem in which there are $2^{\frac{n+2}{3}-1}$ *noncongruent* solutions (a definition is given in Section 2). Section 4 presents a dynamic programming algorithm for SPDP. Algorithm ENUM is adapted to handle SPDP with measurement errors in Section 5. The results of the computer experiments with the developed algorithms are discussed in Section 6. Section 7 contains a brief summary of the results and suggestions for future research.

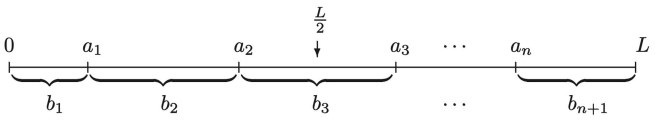

Fig. 1. Graphical interpretation of parameters a_j and b_j.

2 SPDP

Let us discuss a mathematical model for genome mapping based on the simplified partial digest experiment. From the first and second parts of this experiment, we obtain multisets A and B, respectively, of molecule fragment lengths. Let L be the length of the target DNA molecule and let $1, \ldots, n$ be the restriction sites to be recognized by the used enzyme in this molecule. We first assume that the experiment is error free such that multiset A is comprised of n pairs $\{a_j, L - a_j\}$ of positive numbers, where a_j is the length of the fragment including one specified end of the molecule and restriction site j and $L - a_j$ is the length of the complementary fragment, $j = 1, \ldots, n$. Furthermore, multiset B is comprised of $n + 1$ lengths b_j of fragments between every two adjacent points, that is, restriction sites and the ends of the molecule (see the graphical interpretation in Fig. 1).

The error-free *SPDP* can be formulated in terms of number theory as follows: There is an interval $[0, L]$, a positive integer number n, and two multisets A and B of positive integer numbers such that

$$A = \big\{\{a_j, L - a_j\} \mid j = 1, \ldots, n\big\},$$
$$B = \left\{b_j \mid j = 1, \ldots, n+1, \sum_{j=1}^{n+1} b_j = L\right\}.$$

Multiset A contains at most two identical pairs $\{a_j, L - a_j\}$. Identical pairs, if they exist, correspond to the restriction sites that are symmetric with respect to the middle of the molecule (see points a_2 and a_3 in Fig. 1 such that $a_2 = L - a_3$). Each pair $\{a_j, L - a_j\}$ can be ordered as $(a_j, L - a_j)$ or $(L - a_j, a_j)$. Assume that each such ordered pair (p_j, s_j) is associated with a point $p_j \in [0, L]$ so that p_j and $s_j = L - p_j$ are the distances between points 0 and p_j and between points p_j and L, respectively.

The multisets A and B satisfy the property that each pair $\{a_j, L - a_j\}, j = 1, \ldots, n$, can be ordered so that the associated points p_j, $j = 1, \ldots, n$ partition the interval $[0, L]$ into $n + 1$ subintervals with *interpoint distances* constituting the multiset B, that is, $\{p_{j+1} - p_j \mid j = 0, 1, \ldots, n\} = B$, where $p_0 = 0$ and $p_{n+1} = L$.

The problem (SPDP) is concerned with finding a sequence of points $p^* = (0, p_1^*, \ldots, p_n^*, L)$ such that

$$\Big\{\{p_j^*, L - p_j^*\} \mid j = 1, \ldots, n\Big\} = A$$
$$\text{and } \Big\{p_{j+1}^* - p_j^* \mid j = 0, 1, \ldots, n\Big\} = B,$$

where $p_0^* = 0$ and $p_{n+1}^* = L$.

Notice that SPDP always has a solution (the one that corresponds to the original DNA). Furthermore, it may have several solutions. One can be interested in finding one, several, or all mutually noncongruent

solutions of SPDP. Two solutions, $(0, p_1, p_2, \ldots, p_n, L)$ and $(0, p'_1, p'_2, \ldots, p'_n, L)$, of SPDP are *congruent* if and only if $(p_1, \ldots, p_n) = (L - p'_1, L - p'_2, \ldots, L - p'_n)$, that is, they are mirror images of each other.

In the real partial digest experiment, different types of experimental errors may appear. This question will be discussed in detail in Section 5, where the most serious error type—measurement errors—will be modeled and then handled by the use of *interval computations* (for more information about the theory of interval computations, see, for example, Kearfott [17]).

The existing algorithmic results for SPDP include the following: Blazewicz and Kasprzak [4] proved that error-free SPDP is NP-hard in the strong sense and presented an $O(n \log n)$ time algorithm for the case where $b_j \in \{1, 2\}$, $j = 1, \ldots, n+1$. ENUMs for the general error-free SPDP were proposed by Blazewicz et al. [2] and Blazewicz and Jaroszewski [3]. The complexity of SPDP with errors was studied in [6] and [7].

A new $O(n2^n)$ time algorithm that enumerates all mutually noncongruent solutions of SPDP is described in Section 3. An $O(n^{2q})$ time dynamic programming algorithm to find one solution of SPDP, where q is the number of distinct values in the multiset B, is given in Section 3. Later on, the first algorithm (ENUM) is adapted to handle SPDP with measurement errors.

3 An Enumeration of All Noncongruent Solutions

In this section, we present an algorithm, denoted as ENUM, which constructs all mutually noncongruent solutions of the error-free SPDP. The idea of the algorithm is to enumerate all possible locations of the restriction sites based on the distance multisets A and B.

Consider the smallest distance in the multiset A. Let it be a_1. A restriction site corresponding to a_1 is the closest to one of the two ends of the molecule. If there is only one pair $\{a_1, L - a_1\} \in A$ (case 1), then algorithm ENUM constructs a partial solution $p = (0, a_1, \circ, \ldots, \circ, L)$ that contains point $a_1 \in [0, L]$. Here, $\circ$ is the symbol that represents *empty*. Another possibility for the position of the corresponding restriction site is the symmetric point $L - a_1$. However, there is no need to consider solutions containing this point because, for every such solution, there exists a congruent solution containing the point a_1. If there are two identical pairs $\{a_1, L - a_1\} \in A$ (case 2), then algorithm ENUM constructs a partial solution $p = (0, a_1, \circ, \ldots, \circ, L - a_1, L)$ containing two points a_1 and $L - a_1$. The constructed partial solution generates one interpoint distance a_1 (between points 0 and a_1) in case 1 and two interpoint distances a_1 (between points 0 and a_1 and between points $L - a_1$ and L) in case 2. The multiset of unused interpoint distances, that is, $B \setminus \{a_1\}$ in case 1 and $B \setminus \{a_1, a_1\}$ in case 2, is stored with the constructed partial solution p. It is denoted as $B(p)$. Then, the second smallest distance in the multiset A is considered. Let it be a_2. A restriction site corresponding to a_2 is the second closest to one of the two ends of the molecule. Assume that the previously constructed partial solution is $p = (0, a_1, \circ, \ldots, \circ, L)$. If there is one pair $\{a_2, L - a_2\} \in A$, then algorithm ENUM extends p by point a_2 and verifies whether the interpoint distance $a_2 - a_1$ (between points a_1 and a_2) belongs to the multiset $B(p)$. If $a_2 - a_1 \in B(p)$, then the extended solution $p' = (0, a_1, a_2, \circ, \ldots, \circ, L)$ is kept and

$$B(p') \leftarrow B(p) \setminus \{a_2 - a_1\}.$$

If $a_2 - a_1 \notin B(p)$, then p' is discarded. Algorithm ENUM also extends p by the symmetric point $L - a_2$ and verifies whether the interpoint distance a_2 (between points $L - a_2$ and L) belongs to the multiset $B(p)$. If $a_2 \in B(p)$, then the extended solution $p'' = (0, a_1, \circ, \ldots, \circ, L - a_2, L)$ is kept and $B(p'') := B(p) \setminus \{a_2\}$. If $a_2 \notin B(p)$, then p'' is discarded. The process is repeated for n smallest distances in multiset A.

In algorithm ENUM, X_k denotes the set of partial solutions of SPDP which have survived the consideration of the k smallest distances in the multiset A, $k = 1, \ldots, n$. Each partial solution from X_k is a sequence of points $(0, p_1, \ldots, p_s, \circ, \ldots, \circ, p_t, \ldots, p_n, L)$ such that k points $p_1, \ldots, p_s$ and $p_t, \ldots, p_n$, where

$$0 < p_1 < \ldots < p_s < p_t < \ldots < p_n < L,$$

$s + n - t + 1 = k$, are determined and the remaining $n - k$ points between points p_s and p_t are to be determined. The set X_n contains all mutually noncongruent solutions of SPDP. As mentioned above, with each partial solution $p = (0, p_1, \ldots, p_s, \circ, \ldots, \circ, p_t, \ldots, p_n, L) \in X_k$, we associate multiset $B(p)$ of interpoint distances to be used for its extension to a complete solution of SPDP:

$$B(p) = B \setminus \{p_j - p_{j-1} \mid j = 1, \ldots, s, \\ j = t+1, \ldots, n+1\}, \ p_0 := 0, \ p_{n+1} := L.$$

The algorithm can be outlined as follows:

Algorithm ENUM

Step 1. In the multiset A, detect all pairs $\{a_j, L - a_j\}$, $j = 1, \ldots, n$. Order each pair $\{a_j, L - a_j\}$ so that the corresponding ordered pair $(o_j, L - o_j)$ satisfies $o_j \le L - o_j$, $j = 1, \ldots, n$. Renumber the ordered pairs such that $o_1 \le \cdots \le o_n$. Set $X_0 = \{(0, \circ, \ldots, \circ, L)\}$, $B(0, \circ, \ldots, \circ, L) = B$ and $j = 0$.

Step 2. There are four mutually exclusive cases to consider.

Case 1. $j \le n - 3$ and $o_{j+1} = o_{j+2}$. In this case, compute

$$X_{j+2} = \\ \{(0, p_1, \ldots, p_s, o_{j+1}, \circ, \ldots, \circ, L - o_{j+2}, p_t, \ldots, p_n, L) \mid \\ p' = (0, p_1, \ldots, p_s, \circ, \ldots, \circ, p_t, \ldots, p_n, L) \in X_j, \\ o_{j+1} - p_s \in B(p'), \quad p_t - L + o_{j+2} \in B(p')\}.$$

For each $p \in X_{j+2}$ obtained from $p' \in X_j$, calculate $B(p) = B(p') \setminus \{o_{j+1} - p_s, p_t - L + o_{j+2}\}$. Reset $j := j + 2$ and repeat Step 2.

Case 2. $j = n - 2$ and $o_{j+1} = o_{j+2}$. Compute

$$X_n = \{(0, p_1, \ldots, p_s, o_{n-1}, L - o_n, p_t, \ldots, p_n, L) \mid \\ p' = (0, p_1, \ldots, p_s, \circ, \circ, p_t, \ldots, p_n, L) \in X_{n-2}, \\ o_{n-1} - p_s \in B(p'), \quad p_t - L + o_n \in B(p'), \\ L - (o_{n-1} + o_n) \in B(p')\}$$

and stop.

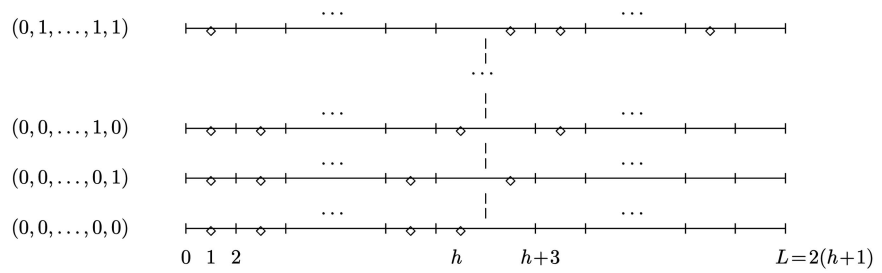

Fig. 2. $2^{\frac{h+1}{2}-1} = 2^{\frac{n+2}{3}-1}$ noncongruent solutions of SPDP.

Case 3. $j \leq n-2$ and $o_{j+1} \neq o_{j+2}$. If this case occurs for the first time, then set X_j contains a single partial solution $p' = (0, p_1, \ldots, p_s, \circ, \ldots, \circ, p_{n-s+1}, \ldots, p_n, L)$ with points symmetric with respect to the middle of the interval $[0, L]$. If Case 3 occurs for the first time, in order to avoid congruent solutions in the set X_n, we calculate

$$X_{j+1} = \{(0, p_1, \ldots, p_s, o_{j+1}, \circ, \ldots, \circ, p_{n-s+1}, \ldots, p_n, L)\}$$

containing one partial solution and

$$B(p) = B(p') \setminus \{o_{j+1} - p_s\}.$$

We do not need to check $o_{j+1} - p_s \in B(p')$ because error-free SPDP must have at least one solution.

For other occurrences of Case 3, we calculate

$$\begin{aligned} X_{j+1} = \Big\{ & p = (0, p_1, \ldots, p_s, o_{j+1}, \circ, \ldots, \circ, p_t, \ldots, p_n, L) \mid \\ & p' = (0, p_1, \ldots, p_s, \circ, \ldots, \circ, p_t, \ldots, p_n, L) \in X_j, \\ & o_{j+1} - p_s \in B(p') \Big\} \cup \\ \Big\{ & \bar{p} = (0, p_1, \ldots, p_s, \circ, \ldots, \circ, L - o_{j+1}, p_t, \ldots, p_n, L) \mid \\ & p' = (0, p_1, \ldots, p_s, \circ, \ldots, \circ, p_t, \ldots, p_n, L) \in X_j, \\ & p_t - L + o_{j+1} \in B(p') \Big\}. \end{aligned}$$

For each $p \in X_{j+1}$ obtained from $p' \in X_j$, calculate $B(p) = B(p') \setminus \{(o_{j+1} - p_s)\}$ and, for each $\bar{p} \in X_{j+1}$ obtained from $p' \in X_j$, calculate $B(\bar{p}) = B(p') \setminus \{(p_t - L + o_{j+1})\}$. Reset $j := j+1$ and repeat Step 2.

Case 4. $j = n-1$. If Case 3 never occurred, then calculate

$$\begin{aligned} X_n = \{ & (0, p_1, \ldots, p_s, o_n, p_t, \ldots, p_n, L) \mid \\ & (0, p_1, \ldots, p_s, \circ, p_t, \ldots, p_n, L) \in X_{n-1} \} \end{aligned}$$

and stop.

If Case 3 occurred at least once, then calculate

$$\begin{aligned} X_n = \Big\{ & (0, p_1, \ldots, p_s, o_n, p_t, \ldots, p_n, L) \mid \\ & p' = (0, p_1, \ldots, p_s, \circ, p_t, \ldots, p_n, L) \in X_{n-1}, \\ & o_n - p_s \in B(p'), \quad p_t - o_n \in B(p') \Big\} \cup \\ & \Big\{ (0, p_1, \ldots, p_s, L - o_n, p_t, \ldots, p_n, L) \mid \\ & p' = (0, p_1, \ldots, p_s, \circ, p_t, \ldots, p_n, L) \in X_{n-1}, \\ & L - o_n - p_s \in B(p'), \quad p_t - L + o_n \in B(p') \Big\} \end{aligned}$$

and stop.

It is clear that point o_j or point $L - o_j$ must be present in every solution of SPDP for each $j = 1, \ldots, n$. If $o_j = o_{j+1}$, then $(o_j, L - o_j) = (o_{j+1}, L - o_{j+1})$ and, hence, both points o_j and o_{j+1} must be present in every solution of SPDP. Algorithm ENUM enumerates all such solutions and removes those with interpoint distances not from the multiset B. Therefore, the algorithm is correct. Step 2 determines the time complexity of the algorithm. In iteration $j+1$ of this step, at most $2|X_j|$ sequences $(0, p_1, \ldots, p_n, L)$ are analyzed. Each sequence is analyzed in a constant time and it can be written to the set X_{j+1} or the set X_{j+2} in $O(n)$ time (if writing each element of a sequence requires one operation). We have $|X_{j+1}| \leq 2|X_j|$, $j = 0, 1, \ldots, n-1$. Since $|X_0| = 1$, we obtain $|X_j| \leq 2^j$, $j = 1, \ldots, n$. The time complexity of Step 2 can be evaluated as $O(n \sum_{j=1}^{n} |X_j|) = O(n2^n)$, which is also the time complexity of ENUM. Its space requirement is determined by $n \max_{1 \leq i \leq j \leq n} \{|X_i| + |X_j|\}$. It can also be evaluated as $O(n2^n)$.

We now give an example of SPDP for which there are $2^{\frac{n+2}{3}-1}$ noncongruent solutions, that is, the cardinality of the set X_n is exponential in this case. Our example partially answers a question about the number of solutions of SPDP posed by Blazewicz et al. [1].

Let n and h be positive integer numbers satisfying $h + (h+1)/2 - 1 = n$. In our example, the length of the interval is $L = 2(h+1)$, multiset

$$\begin{aligned} A = & \Big\{ \{2j-1, L-(2j-1)\} \mid j = 1, \ldots, (h+1)/2 \Big\} \\ & \cup \Big\{ \{2j, L-2j\} \mid j = 1, \ldots, \frac{h+1}{2} - 1, \frac{h+1}{2} + 1, \ldots, h \Big\}, \end{aligned}$$

and multiset B consists of h numbers 1, $(h+1)/2 - 1$ numbers 2, and one number 3. Noncongruent solutions for this example are shown in Fig. 2.

In the figure, the 0-1 vector $(y_1, \ldots, y_{\frac{h+1}{2}})$ is associated with the points "$\diamond$." A point "$\diamond$" associated with y_j can be placed either in the position $2j-1$ or in the symmetric position $L - 2j + 1$. In the former case, $y_j = 0$ and, in the latter case, $y_j = 1$.

Denote by $Y_{\frac{h+1}{2}}$ the set of all 0-1 vectors $(y_1, \ldots, y_{\frac{h+1}{2}})$. All noncongruent solutions of SPDP are determined by the set $\hat{Y}_{\frac{h+1}{2}} \subset Y_{\frac{h+1}{2}}$ so that there are no two vectors y and y' in $\hat{Y}_{\frac{h+1}{2}}$ satisfying $y'_j = 1 - y_j$, $j = 1, \ldots, (h+1)/2$. Since $|Y_{\frac{h+1}{2}}| = 2^{\frac{h+1}{2}}$ and $|\hat{Y}_{\frac{h+1}{2}}| = |Y_{\frac{h+1}{2}}|/2$, we obtain $|\hat{Y}_{\frac{h+1}{2}}| = 2^{\frac{h+1}{2}-1} = 2^{\frac{n+2}{3}-1}$.

Our example can easily be transformed into an example in which interpoint distances are all distinct. For these

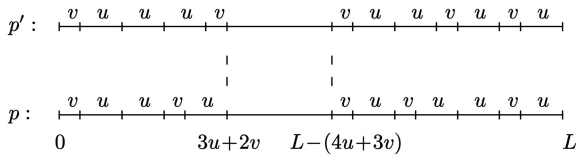

Fig. 3. Two partial solutions in the same state.

purposes, let the picture for the points "|" be symmetric with respect to the middle, but let the distances between any two adjacent such points be all distinct in the interval $[0, L/2] = [0, h-1]$. Let every point "$\diamond$" partition an interval between two adjacent points "|" so that all generated distances are distinct. Furthermore, let the two interpoint distances between the points h, "$\diamond$," and $h+3$ be distinct and differ from all other interpoint distances. As with the original example, every point "$\diamond$" can be placed in one of the two symmetric positions associated with it. Therefore, there are $2^{\frac{n+2}{3}-1}$ noncongruent solutions for this example.

In Section 4, a dynamic programming algorithm will be presented for a special case of the error-free SPDP. It will be tested against ENUM in an extensive computational experiment in Section 6. Algorithm ENUM will be modified to cover the case of measurement errors (cf., Section 5 and tests in Section 6).

4 A Dynamic Programming Algorithm

Dynamic programming is a well-established search technique that has grown out of the operational research tradition. Other complementary search procedures can be seen in [5]. A basic introduction to the technique can be found in the tutorial by Dowsland [12] and its importance as a method to underpin the development of computationally intelligent systems is presented in the work of Poole et al. [23]. They say, in the context of Computational Intelligence, that dynamic programming "deserves attention because it is important in many optimization problems, particularly those involving decision making."

We first consider a special case of the error-free SPDP in which $b_j \in \{u, v\}$, $j = 1, \ldots, n+1$. We denote this special case as SPDP(u, v) and present a dynamic programming algorithm, denoted as DP, which constructs a solution of SPDP(u, v) in $O(n^4)$ time. We stress that algorithm DP constructs one solution, not all mutually noncongruent solutions of SPDP. We use the same terminology and notations as in the algorithm ENUM.

The idea of the algorithm DP is given as follows: Consider two partial solutions $p \in X_j$ and $p' \in X_j$, see an example in Fig. 3.

Assume that the number of interpoint distances of each type u and v is the same in the left part and in the right part of both partial solutions. Then, 1) the points closest to the middle of the molecule are the same in p and p', and 2) the interpoint distances to be used for the extensions of p and p' are the same: $B(p) = B(p')$. Partial solutions from the same set X_j, which satisfy properties 1 and 2, are said to be in the same *state*. Properties 1 and 2 imply that, if one of the two partial solutions p and p' can be extended to a complete solution of SPDP(u, v), then the other partial solution can do the same. Therefore, if there are several partial solutions in the same state, then only one of them (arbitrary) can be considered for further expansion and all others can be discarded. This selection procedure reduces the cardinality of the set X_j. A more detailed description of algorithm DP is given below.

Assume that the ordered pairs $(o_j, L - o_j)$ of elements from the multiset A satisfy $o_j \le L - o_j$, $j = 1, \ldots, n$, and $o_1 \le \cdots \le o_n$. In the algorithm DP, we assign points associated with the distances $o_1, \ldots, o_n$ to partial solutions of SPDP(u, v) in this order. Given a partial solution $(0, p_1, \ldots, p_s, \circ, \ldots, \circ, p_t, \ldots, p_n, L) \in X_j$, there are two possible assignments of points corresponding to the distance o_{j+1}: Point o_{j+1} is assigned to the position $s+1$ or point $L - o_{j+1}$ is assigned to the position $t-1$. If $o_{j+1} = o_{j+2} = o$, then point o is assigned to the position $s+1$ and point $L - o$ is assigned to the position $t-1$.

Any feasible assignment described above can be viewed as an assignment of at most two interpoint distances, u and v, to the left part and to the right part of a partial solution. To facilitate description of our algorithm, we introduce an operation $LR(x_1, x_2)$ that denotes an assignment of the interpoint distances x_1 and x_2 to the left part and to the right part of a partial solution, respectively. If $x_1 = \circ$ or $x_2 = \circ$, then nothing is assigned to the left part or to the right part, respectively, of the solution under consideration.

A state (j, l_u, l_v, r_u) is associated with each partial solution $(0, p_1, \ldots, p_s, \circ, \ldots, \circ, p_t, \ldots, p_n, L) \in X_j$. *State variables* are defined as follows:

- j is the number of distances o_i, $i = 1, \ldots, j$, considered so far. These distances generate the same number j of interpoint distances between the corresponding points.
- $l_u = |\{p_j - p_{j-1} = u \mid j = 1, \ldots, s\}|$ is the number of interpoint distances equal to u in the left part of the partial solution, where $p_0 = 0$.
- $l_v = |\{p_j - p_{j-1} = v \mid j = 1, \ldots, s\}|$ is the number of interpoint distances equal to v in the left part of the partial solution.
- $r_u = |\{p_j - p_{j-1} = u \mid j = t+1, \ldots, n+1\}|$ is the number of interpoint distances equal to u in the right part of the partial solution, where $p_{n+1} = L$.

Having j, l_u, l_v, and r_u, we can calculate:

- $r_v = |\{p_j - p_{j-1} = v \mid j = t+1, \ldots, n+1\}|$, which is the number of interpoint distances equal to v in the right part of the partial solution. We can compute $r_v = j - (l_u + l_v + r_u)$.

In algorithm DP, we iteratively generate sets S_j of states (j, l_u, l_v, r_u), $j = 1, \ldots, n$. Set S_n is guaranteed to contain a state corresponding to a complete solution of SPDP(u, v). A complete solution is recovered by backtracking. Denote $k_u = |\{j \mid b_j = u, j = 1, \ldots, n+1\}|$ and $k_v = n + 1 - k_u$. Assume, without loss of generality, that $k_u \le k_v$. The algorithm can be outlined as follows.

Algorithm DP

Step 1. In the multiset A, detect all pairs $\{a_j, L - a_j\}$, $j = 1, \ldots, n$. Order each pair $\{a_j, L - a_j\}$ so that the corresponding ordered pair $(o_j, L - o_j)$ satisfies $o_j \le L - o_j$, $j = 1, \ldots, n$. Renumber the ordered pairs

such that $o_1 \leq \cdots \leq o_n$. Initiate set $S_0 = \{(0,0,0,0)\}$ and $j = 0$.

Step 2. For each state $(j, l_u, l_v, r_u) \in S_j$, perform the following computations: Calculate $r_v = j - (l_u + l_v + r_u)$, $T_l = l_u u + l_v v$, and $T_r = r_u u + r_v v$.

With each state $(j, l_u, l_v, r_u) \in S_j$, for $j \geq 1$, associate an *indicator variable* $J \in \{1, \ldots, 12\}$ to be used for the purposes of recovering a complete solution corresponding to a final state (n, l_u, l_v, r_u). Its meaning will be clear from our description.

There are four mutually exclusive cases to consider:

Case 1. $j \leq n-3$ and $o_{j+1} = o_{j+2}$. In this case, initiate set $S_{j+2} = \circ$.

If $o_{j+1} - T_l = u$, $o_{j+2} - T_r = u$, and $l_u + r_u + 2 \leq k_u$, then add state $(j+2, l_u + 1, l_v, r_u + 1)$ to the set S_{j+2}. With this state, associate the value $J = 1$, indicating that it was obtained by performing operation $LR(u,u)$ over the corresponding partial solution from the set S_j.

If $o_{j+1} - T_l = u$, $o_{j+2} - T_r = v$, $l_u + r_u + 1 \leq k_u$, and $l_v + r_v + 1 \leq k_v$, then add state $(j+2, l_u + 1, l_v, r_u)$ to S_{j+2}. With this state, associate the value $J = 2$ identifying operation $LR(u,v)$.

If $o_{j+1} - T_l = v$, $o_{j+2} - T_r = u$, $l_u + r_u + 1 \leq k_u$, and $l_v + r_v + 1 \leq k_v$, then add state $(j+2, l_u, l_v + 1, r_u + 1)$ to S_{j+2}. With this state, associate the value $J = 3$ identifying operation $LR(v,u)$.

If $o_{j+1} - T_l = v$, $o_{j+2} - T_r = v$, and $l_v + r_v + 2 \leq k_v$, then add state $(j+2, l_u, l_v + 1, r_u)$ to S_{j+2}. With this state, associate the value $J = 4$ identifying operation $LR(v,v)$.

Case 2. $j \leq n-2$ and $o_{j+1} \neq o_{j+2}$. In this case, initiate set $S_{j+1} = \circ$.

If $o_{j+1} - T_l = u$ and $l_u + r_u + 1 \leq k_u$, then add state $(j+1, l_u + 1, l_v, r_u)$ to S_{j+1}. With this state, associate the value $J = 5$ identifying operation $LR(u, \circ)$.

If $o_{j+1} - T_l = v$ and $l_v + r_v + 1 \leq k_v$, then add state $(j+1, l_u, l_v + 1, r_u)$ to S_{j+1}. With this state, associate the value $J = 6$ identifying operation $LR(v, \circ)$.

If $o_{j+1} - T_r = u$ and $l_u + r_u + 1 \leq k_u$, then add state $(j+1, l_u, l_v, r_u + 1)$ to S_{j+1}. With this state, associate the value $J = 7$ identifying operation $LR(\circ, u)$.

If $o_{j+1} - T_r = v$ and $l_v + r_v + 1 \leq k_v$, then add state $(j+1, l_u, l_v, r_u)$ to S_{j+1}. With this state, associate the value $J = 8$ identifying operation $LR(\circ, v)$.

Case 3. $j = n-2$ and $o_{j+1} = o_{j+2}$. In this case, initiate set $S_n = \phi$.

If $o_{n-1} - T_l = u$, $L - (o_{n-1} + o_n) = u$, $L - T_r - o_n = u$, and $l_u + r_u + 3 = k_u$, then add state $(n, l_u + 3, l_v, r_u)$ to S_n. With this state, associate the value $J = 1$. We can use the same values of the indicator variable as in Cases 1 and 2 because $j = n$ and $j < n$ are obviously distinguished by the backtracking procedure described in Step 3.

If $o_{n-1} - T_l = u$, $L - (o_{n-1} + o_n) = u$, $L - T_r - o_n = v$, and $l_u + r_u + 2 = k_u$, then add state $(n, l_u + 2, l_v + 1, r_u)$ to S_n. With this state, associate the value $J = 2$.

If $o_{n-1} - T_l = u$, $L - (o_{n-1} + o_n) = v$, $L - T_r - o_n = u$, and $l_u + r_u + 2 = k_u$, then add state $(n, l_u + 2, l_v + 1, r_u)$ to S_n. With this state, associate the value $J = 3$.

If $o_{n-1} - T_l = u$, $L - (o_{n-1} + o_n) = v$, $L - T_r - o_n = v$, and $l_u + r_u + 1 = k_u$, then add state $(n, l_u + 1, l_v + 2, r_u)$ to S_n. With this state, associate the value $J = 4$.

If $o_{n-1} - T_l = v$, $L - (o_{n-1} + o_n) = u$, $L - T_r - o_n = u$, and $l_u + r_u + 2 = k_u$, then add state $(n, l_u + 2, l_v + 1, r_u)$ to S_n. With this state, associate the value $J = 5$.

If $o_{n-1} - T_l = v$, $L - (o_{n-1} + o_n) = u$, $L - T_r - o_n = v$, and $l_u + r_u + 1 = k_u$, then add state $(n, l_u + 1, l_v + 2, r_u)$ to S_n. With this state, associate the value $J = 6$.

If $o_{n-1} - T_l = v$, $L - (o_{n-1} + o_n) = v$, $L - T_r - o_n = u$, and $l_u + r_u + 1 = k_u$, then add state $(n, l_u + 1, l_v + 2, r_u)$ to S_n. With this state, associate the value $J = 7$.

If $o_{n-1} - T_l = v$, $L - (o_{n-1} + o_n) = v$, $L - T_r - o_n = v$, and $l_u + r_u = k_u$, then add state $(n, l_u, l_v + 3, r_u)$ to S_n. With this state, associate the value $J = 8$.

Case 4. $j = n-1$. In this case, initiate set $S_n = \phi$.

If $o_n - T_l = u$, $L - T_r - o_n = u$, and $l_u + r_u + 2 = k_u$, then add state $(n, l_u + 1, l_v, r_u + 1)$ to S_n. With this state, associate the value $J = 9$. Here, we need a new value of the indicator variable to distinguish between Cases 3 and 4 when considering a final state from the set S_n in the backtracking procedure.

If $o_n - T_l = u$, $L - T_r - o_n = v$, and $l_u + r_u + 1 = k_u$, then add state $(n, l_u + 1, l_v, r_u)$ to S_n. With this state, associate the value $J = 10$.

If $o_n - T_l = v$, $L - T_r - o_n = u$, and $l_u + r_u + 1 = k_u$, then add state $(n, l_u, l_v + 1, r_u + 1)$ to S_n. With this state, associate the value $J = 11$.

If $o_n - T_l = v$, $L - T_r - o_n = v$, and $l_u + r_u = k_u$, then add $(j, l_u, l_v + 1, r_u)$ to S_n. With this state, associate the value $J = 12$.

If $L - o_n - T_l = u$, $o_n - T_r = u$, and $l_u + r_u + 2 = k_u$, then add state $(n, l_u + 1, l_v, r_u + 1)$ to S_n. With this state, associate the value $J = 9$.

If $L - o_n - T_l = u$, $o_n - T_r = v$, and $l_u + r_u + 1 = k_u$, then add state $(n, l_u + 1, l_v, r_u)$ to S_n. With this state, associate the value $J = 10$.

If $L - o_n - T_l = v$, $o_n - T_r = u$, and $l_u + r_u + 1 = k_u$, then add state $(n, l_u, l_v + 1, r_u + 1)$ to S_n. With this state, associate the value $J = 11$.

If $L - o_n - T_l = v$, $o_n - T_r = v$, and $l_u + r_u = k_u$, then add $(j, l_u, l_v + 1, r_u)$ to S_n. With this state, associate the value $J = 12$.

If all states from S_j are considered, reset $j := j + 1$. If $j = n$, then perform Step 3. Otherwise, repeat Step 2.

Step 3. Select any state $(n, l_u, l_v, r_u) \in S_n$ and backtrack to determine the corresponding solution of SPDP(u,v).

In the first iteration of the backtracking procedure, if $J = 1$ is associated with a state $(n, l_u, l_v, r_u) \in S_n$, then we know that this state was obtained from the state $(n-2, l_u - 3, l_v, r_u) \in S_{n-2}$ by assigning three interpoint distancs u to the left part of the corresponding partial solution. Other values of J are similarly analyzed.

In iteration $i \geq 2$ of the backtracking procedure, if $J = 1$ is associated with a state $(i, l_u, l_v, r_u) \in S_i$, then we know that this state was obtained from the state $(i-2, l_u - 1, l_v, r_u - 1) \in S_{i-2}$ by assigning interpoint distances u to the left part and the right part of the corresponding partial solution. Other values of J are similarly analyzed. The procedure terminates when a complete solution of SPDP(u,v) is recovered.

Similarly to algorithm ENUM, Step 2 determines the time complexity of algorithm DP. The time complexity of

this step can be evaluated as $O(\sum_{j=1}^{n}|S_j|)$. Since $l_u \le k_u$, $l_v \le k_v$, and $r_u \le k_u$, we have $|S_j| \le O(k_u^2 k_v)$, $j = 1, \ldots, n$. Therefore, Step 2 requires $O(nk_u^2k_v) = O(n^4)$ operations, giving us the overall time complexity of algorithm DP. Its space requirement is determined by $\sum_{j=1}^{n}|S_j|$ because we store the indicator variable J with each state. Thus, the space requirement of algorithm DP is also $O(n^4)$. Notice that $n^4 \le 2^n$ for $n \ge 16$. Therefore, algorithm DP should be much more efficient than ENUMs with runtime $O(2^n)$ for $\mathrm{SPDP}(u, v)$ with a large number of restriction sites.

Algorithm DP is justified by the following theorem:

Theorem 1. *If state (j, l_u, l_v, r_u) is associated with a partial solution that can be extended to a complete solution of $\mathrm{SPDP}(u, v)$, then any partial solution in this state can be extended to a complete solution of $\mathrm{SPDP}(u, v)$.*

Proof. Assume that a partial solution $p = (0, p_1, \ldots, p_s, \circ, \ldots, \circ, p_t, \ldots, p_n, L) \in X_j$ associated with state $(j, l_u, l_v, r_u) \in S_j$ can be extended to a complete solution $\bar{p} = (0, p_1, \ldots, p_n, L) \in X_n$ of problem $\mathrm{SPDP}(u, v)$ by assigning distances $p_{s+1}, \ldots, p_{t-1}$. Consider another partial solution $p' = (0, p'_1, \ldots, p'_{s'}, \circ, \ldots, \circ, p'_{t'}, \ldots, p'_n, L) \in X_j$ in the same state (j, l_u, l_v, r_u). Notice that $s' = s = l_u + l_v$ and $t' = t = r_u + r_v$.

For both partial solutions p and p', the corresponding values T_l and T_r are the same. Furthermore, the same number of interpoint distances u and the same number of interpoint distances v are present in both solutions. Therefore, partial solution p' can be extended to a complete solution $\bar{p'} = (0, p'_1, \ldots, p'_s, p_{s+1}, \ldots, p_{t-1}, p'_t, \ldots, p'_n, L) \in X_n$ in the same way as p. In this extension, new assignments will generate the same multiset of interpoint distances as in the extension of the partial solution p. Hence, if $\bar{p}$ is a solution of $\mathrm{SPDP}(u, v)$, then $\bar{p'}$ is a solution of this problem as well. □

It is clear that partial solution $(0, \circ, \ldots, \circ, L) \in X_0$ associated with the state $(0, 0, 0, 0) \in S_0$ can be extended to a complete solution of $\mathrm{SPDP}(u, v)$. Then, we can iteratively apply Theorem 1 to demonstrate that algorithm DP finds a solution of $\mathrm{SPDP}(u, v)$.

Algorithm DP can be modified to solve error-free SPDP with any number q of distinct interpoint distances b_j, $j = 1, \ldots, n+1$, which we denote as $\mathrm{SDP}(u_1, \ldots, u_q)$. Here, $u_1, \ldots, u_q$ are all distinct values in the multiset B.

In the modified algorithm, a state $(j, l_1, \ldots, l_q, r_1, \ldots, r_{q-1})$ with $2q$ state variables will be associated with a partial solution. Here, j is the number of smallest distances in the multiset A considered so far. Also, l_i and r_i are the numbers of interpoint distances equal to u_i in the left part and the right part of the partial solution, respectively. We can calculate $r_q = j - (l_1 + \ldots + l_q + r_1 + \ldots + r_{q-1})$.

The time and space requirements of the modified algorithm can be evaluated as $O(n^{2q})$ for a given q. They are polynomial if q is a constant. In the modified algorithm, we will need $q^3 + q^2$ different values of the indicator variable because, in Case 3, each of the three last interpoint distances (left, middle, and right) can take any of the q values b_j and, in Case 4, each of the two last interpoint distances can take any of these values.

5 Measurement Errors

In the literature, three main types of errors are discussed, which can affect the input data for a digesting method; see, for example, Dix and Kieronska [11], Inglehart and Nelson [16], and Wright et al. [29]. Errors of the first type are caused by the *imprecise measurement* of the lengths of the cut fragments. According to Marra et al. [21], the relative deviation of 5 percent from the true fragment length can be guaranteed in a biochemical experiment (it was within 1.5 percent for 95 percent of fragments with lengths between 600 and 12,000 base pairs in a specific experiment). Errors of the second type, which are called *negative errors*, are due to the loss of information about some fragments. Recall that the length determination experiment is conducted on a sufficiently large number of DNA clones and a particular fragment length is accepted if the quantity of the corresponding fragments exceeds a given threshold. A negative error can occur when there are two fragments of almost the same length such that they cannot be distinguished in the experiment. It can also occur when some small fragments are lost because they are too speedy and cannot be traced. Finally, certain sites can be less likely to be cut than others and the quantity of the corresponding fragments can be insufficient for accepting their lengths. Errors of the third type, called *positive errors*, appear when an enzyme erroneously cuts a DNA molecule at a place that is similar to but not the same as the restriction site associated with this enzyme. They can also appear when some unrelated material is occasionally added to the gel. In this case, we may obtain some fragments that do not belong to the target DNA. Moreover, a clone can be cut in more than one restriction site in the short digestion reaction.

When an error of the second or third type occurs in the simplified partial digest experiment, it can be identified and sometimes corrected as follows: If there is a large length a_j and no small complementary length $L - a_j$ in the multiset A, then we can deduce that the fragment of the length $L - a_j$ was lost in the experiment. If there are more than two identical pairs $\{a_j, L - a_j\}$ in the multiset A, then we know that only two of them can correspond to the symmetric restriction sites and the other pairs are obtained erroneously. Several positive errors can be recognized during the phase of gel electrophoresis, where the fragment lengths are measured—simply, the erroneous fragments will occur in negligible amounts.

Measurement errors of the first type are difficult to avoid in any digesting experiment. In this section, we describe an adaptation of algorithm ENUM for the case when distances in the multisets A and B are measured with the guaranteed relative error r. Then, this algorithm will construct an approximate solution with interpoint distances being within a δ distance from corresponding interpoint distances of an error-free SPDP solution. If an instance of error-free SPDP has more than one feasible solution, the algorithm will find one approximate solution for every feasible solution of this instance.

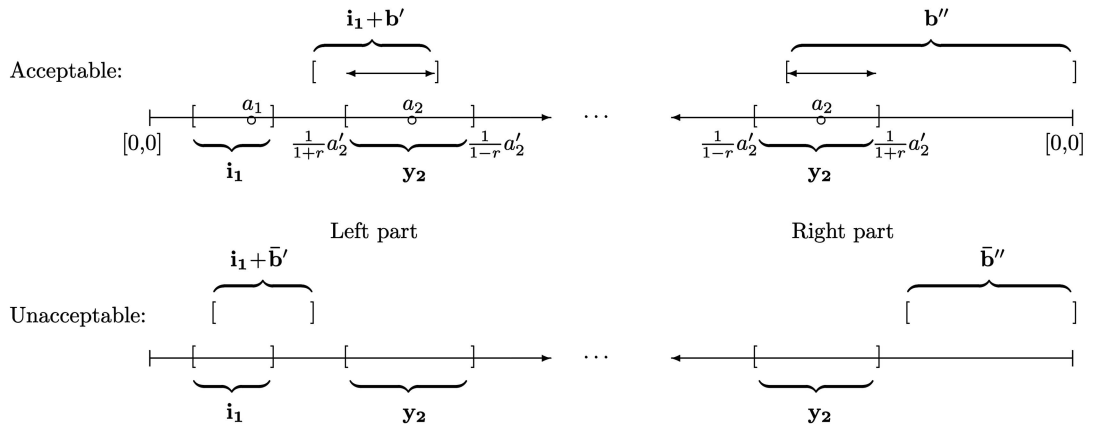

Fig. 4. Assignments of the interval $\mathbf{y_2} = [\frac{1}{1+r}a'_2, \frac{1}{1-r}a'_2]$.

Specifically, we assume that there are given multisets $A' = \{a'_j \mid j = 1, \ldots, 2n\}$ and $B' = \{b'_j \mid j = 1, \ldots, n+1, \sum_{j=1}^{n+1} b'_j = L'\}$ such that there exists a DNA molecule with a multiset $A = \{a_j \mid j = 1, \ldots, 2n\}$ of true distances between the restriction sites and the two ends of the molecule and a multiset $B = \{b_j \mid j = 1, \ldots, n+1, \sum_{j=1}^{n+1} b_j = L\}$ of true interpoint distances, which satisfy

$$\frac{|a'_j - a_j|}{a_j} \le r,\ j = 1, \ldots, 2n, \quad \text{and } \frac{|b'_j - b_j|}{b_j} \le r,\ j = 1, \ldots, n+1 \tag{1}$$

for a given relative error r, $0 < r < 1$. Notice that there may exist several distinct molecules satisfying this property for the same multisets A' and B'.

Let P^* be the set of all mutually noncongruent solutions of the error-free SPDP with the multisets A and B, where each $p^* \in P^*$ is a sequence of increasing points in the interval $[0, L]$, $p^* = (p^*_0, p^*_1, \ldots, p^*_n, p^*_{n+1})$, $p^*_0 = 0$, and $p^*_{n+1} = L$.

Given multisets A' and B' and a point sequence $p^* \in P^*$, a (δ, p^*)-*approximate* solution to the problem (SPDP) *with measurement errors* is a point sequence $p' = (p'_0, p'_1, \ldots, p'_n, p'_{n+1})$ such that

$$\max\left\{\frac{|p'_j - p^*_j|}{p^*_j}, \frac{|(p'_{n+1} - p'_j) - (p^*_{n+1} - p^*_j)|}{p^*_{n+1} - p^*_j}, \frac{|(p'_{j+1} - p'_j) - (p^*_{j+1} - p^*_j)|}{p^*_{j+1} - p^*_j}\right\} \le \delta,\ j = 0, 1, \ldots, n.$$

For SPDP with measurement errors, we adapt algorithm ENUM to work with intervals rather than with single numbers. Given relative error r, an interval $[\frac{1}{1+r}a'_j, \frac{1}{1-r}a'_j]$ is associated with each $a'_j \in A'$ and an interval $[\frac{1}{1+r}b'_j, \frac{1}{1-r}b'_j]$ is associated with each $b'_j \in B'$. This choice of the intervals guarantees that if $a_j \in A$ and $b_j \in B$ are true interpoint distances satisfying (1), then $a_j \in [\frac{1}{1+r}a'_j, \frac{1}{1-r}a'_j]$ and $b_j \in [\frac{1}{1+r}b'_j, \frac{1}{1-r}b'_j]$, that is, the true interpoint distances are inside the chosen intervals.

We denote the adapted algorithm as Interval-ENUM. In this algorithm, we use the operation of summation of two intervals $\mathbf{i} = [a, b]$, $a \le b$, and $\mathbf{g} = [c, d]$, $c \le d$, which is defined as $\mathbf{i} + \mathbf{g} = [a + c, b + d]$. This definition is in accordance with the theory of interval computations; see, for example, Kearfott [17].

Algorithm Interval-ENUM constructs a $(\frac{2r}{1-r}, p^*)$-approximate solution for each $p^* \in P^*$. As in algorithm ENUM, we construct sets X'_k, $k = 1, \ldots, n$, of partial *interval solutions*, where a partial interval solution from X'_k is a sequence of intervals $\mathbf{i} = (\mathbf{i_0}, \mathbf{i_1}, \ldots, \mathbf{i_s}, \circ, \ldots, \circ, \mathbf{i_t}, \ldots, \mathbf{i_{n+1}})$, $s + n - t + 1 = k$, $\mathbf{i_0} = \mathbf{i_{n+1}} = [0, 0]$. Let us introduce multiset $I := \{[\frac{1}{1+r}b, \frac{1}{1-r}b] \mid b \in B'\}$. With each partial solution $\mathbf{i}$, a multiset of unused intervals $I(\mathbf{i}) \subset I$ is stored whose precise definition will be clear from the description of the algorithm.

Let $a'_1 \le \cdots \le a'_{2n}$. In the first iteration of algorithm Interval-ENUM, we assign the interval $\mathbf{y_1} := [\frac{1}{1+r}a'_1, \frac{1}{1-r}a'_1]$ to the left part of the molecule and find a multiset B_1 of $\mathbf{b} \in I$ such that $\mathbf{i}_1(\mathbf{b}) := (\mathbf{i_0} + \mathbf{b}) \cap \mathbf{y_1} = \mathbf{b} \cap \mathbf{y_1} \ne \phi$. We set $X'_1 = \{(\mathbf{i_0}, \mathbf{i_1}(\mathbf{b}), \circ, \ldots, \circ, \mathbf{i_{n+1}}) \mid \mathbf{b} \in B_1\}$. Thus, if there exists a partial solution $p = (0, a_1, \circ, \ldots, \circ, L)$ to the error-free SPDP with distance multisets A and B, then $a_1 \in \mathbf{i_1}(\mathbf{b})$ for $\mathbf{b} \in B_1$. We do not consider assignment of the interval $\mathbf{y_1}$ to the right part of the molecule to avoid congruent solutions. Furthermore, since we work with imprecise data, we do not separately consider the case of several identical intervals $\mathbf{y_1}$.

In the second iteration, we consider assignments of the interval $\mathbf{y_2} := [\frac{1}{1+r}a'_2, \frac{1}{1-r}a'_2]$ to the left part and to the right part of the molecule for each $\mathbf{i} = (\mathbf{i_0}, \mathbf{i_1}, \circ, \ldots, \circ, \mathbf{i_{n+1}}) \in X'_1$. An assignment to the left part is accepted if $(\mathbf{i_1} + \mathbf{b}') \cap \mathbf{y_2} \ne \phi$ for some $\mathbf{b}' \in I(\mathbf{i})$; see Fig. 4 for a graphical interpretation.

Similarly, an assignment to the right part is accepted if $(\mathbf{i_{n+1}} + \mathbf{b}'') \cap \mathbf{y_2} = \mathbf{b}'' \cap \mathbf{y_2} \ne \phi$ for some $\mathbf{b}'' \in I(\mathbf{i})$. Here, there is a small difference from algorithm ENUM, namely, the distance to the interval assigned to the right part of the molecule is counted from the right end of the molecule. In algorithm ENUM, the corresponding distance was $L - a_2$, that is, it was counted from the left end of the molecule.

Set X'_n is such that, for each solution $p^* \in P^*$ of the error-free SPDP, there exists a sequence of intervals $\mathbf{i}^* = (\mathbf{i^*_0}, \mathbf{i^*_1}, \ldots, \mathbf{i^*_{n+1}}) \in X'_n$ satisfying the following two properties:

- $p_j^* \in \mathbf{i}_\mathbf{j}^*$ for $j = 0, 1, \ldots, s$ and $L - p_j^* \in \mathbf{i}_\mathbf{j}^*$ for $j = s+1, \ldots, n+1$, where $\mathbf{i}_\mathbf{s}^*$ is the last interval assigned to the left part of the molecule and
- $p_j^* - p_{j-1}^* \in \mathbf{b}'_\mathbf{j}$, $j = 1, \ldots, n+1$, where

$$\{\mathbf{b}'_\mathbf{1}, \ldots, \mathbf{b}'_{\mathbf{n+1}}\} = I.$$

By the construction of intervals $\mathbf{i}_\mathbf{j}^*$, any point sequence $p' = (p'_0, p'_1, \ldots, p'_n, p'_{n+1})$ that satisfies the above properties (when substituting p' instead of p^*) is a $(\frac{2r}{1-r}, p^*)$-approximate solution for SPDP with measurement errors. Such a point sequence p' is constructed for each $\mathbf{i} \in X'_n$ in Step 4 of algorithm Interval-ENUM. The algorithm can be formally described as follows.

Algorithm Interval-ENUM

Step 1. Order numbers in the multiset A' so that $a'_1 \le \cdots \le a'_{2n}$. Introduce intervals $\mathbf{y_j} = [\frac{1}{1+r}a'_j, \frac{1}{1-r}a'_j]$, $j = 1, \ldots, n$, and the multiset of intervals $I = \{[\frac{1}{1+r}b, \frac{1}{1-r}b] \mid b \in B'\}$. Calculate $L' = \sum_{j=1}^{n+1} b'_j$. Set $X'_0 = \{(\mathbf{i_0}, \circ, \ldots, \circ, \mathbf{i_{n+1}})\}$, where $\mathbf{i_0} = \mathbf{i_{n+1}} = [0, 0]$, $B(\mathbf{i_0}, \circ, \ldots, \circ, \mathbf{i_{n+1}}) = I$, and $j = 0$.

Step 2 (the case $j \le n-2$). If $j = n-1$, go to Step 3. Otherwise, perform the following computations. For each partial solution $\mathbf{i}' = (\mathbf{i_0}, \mathbf{i_1}, \ldots, \mathbf{i_s}, \circ, \ldots, \circ, \mathbf{i_t}, \ldots, \mathbf{i_{n+1}}) \in X'_j$, find a multiset B_1 of intervals $\mathbf{b} \in I(\mathbf{i}')$ such that $\mathbf{i_{s+1}}(\mathbf{b}) := (\mathbf{i_s} + \mathbf{b}) \cap \mathbf{y_{j+1}} \ne \phi$ and, if $j \ge 1$, a multiset B_2 of intervals $\mathbf{b} \in I(\mathbf{i}')$ such that $\mathbf{i_{t-1}}(\mathbf{b}) := (\mathbf{i_t} + \mathbf{b}) \cap \mathbf{y_{j+1}} \ne \phi$. With each interval $\mathbf{i_{s+1}}(\mathbf{b})$ and $\mathbf{i_{t-1}}(\mathbf{b})$, store the corresponding interval $\mathbf{b}$ by setting $L(\mathbf{i_{s+1}}(\mathbf{b})) := \mathbf{b}$ and $R(\mathbf{i_{t-1}}(\mathbf{b})) := \mathbf{b}$, respectively.

If $j = 0$, set $B_2 = \phi$. The case $j = 0$ is treated separately to avoid congruent solutions. Calculate

$$X'_{j+1} = \{(\mathbf{i_0}, \mathbf{i_1}, \ldots, \mathbf{i_s}, \mathbf{i_{s+1}}(\mathbf{b}), \circ, \ldots, \circ, \mathbf{i_t}, \ldots, \mathbf{i_{n+1}}) \mid \mathbf{i}' \in X'_j, \mathbf{b} \in B_1\} \cup \{(\mathbf{i_0}, \mathbf{i_1}, \ldots, \mathbf{i_s}, \circ, \ldots, \circ, \mathbf{i_{t-1}}(\mathbf{b}), \mathbf{i_t}, \ldots, \mathbf{i_{n+1}}) \mid \mathbf{i}' \in X'_j, \mathbf{b} \in B_2\}.$$

For each $\mathbf{i} \in X'_{j+1}$ obtained from $\mathbf{i}' \in X'_j$ and $\mathbf{b} \in I(\mathbf{i}')$, calculate $I(\mathbf{i}) = I(\mathbf{i}') \setminus \{\mathbf{b}\}$.

Reset $j := j + 1$ and repeat Step 2.

Step 3 (the case $j = n-1$). Set $X'_n = \phi$. For each partial solution $\mathbf{i}' = (\mathbf{i_0}, \mathbf{i_1}, \ldots, \mathbf{i_s}, \circ, \mathbf{i_t}, \ldots, \mathbf{i_{n+1}}) \in X'_{n-1}$, let $I(\mathbf{i}') = \{\mathbf{b_1}, \mathbf{b_2}\}$.

If $\mathbf{i_{s+1}}(\mathbf{b_1}) := (\mathbf{i_s} + \mathbf{b_1}) \cap \mathbf{y_n} \ne \phi$ and

$$(\mathbf{i_{s+1}}(\mathbf{b_1}) + \mathbf{b_2} + \mathbf{i_t}) \cap \left[\frac{1}{1+r}L', \frac{1}{1-r}L'\right] \ne \phi,$$

then set $X'_n := X'_n \cup \{\mathbf{i} = (\mathbf{i_0}, \mathbf{i_1}, \ldots, \mathbf{i_s}, \mathbf{i_{s+1}}(\mathbf{b_1}), \mathbf{i_t}, \ldots, \mathbf{i_{n+1}})\}$. Store the number $n_L(\mathbf{i}) := s + 1$ of intervals assigned to the left part of the molecule corresponding to $\mathbf{i}$.

The last interval assigned to the left part is specific in that we store the interval from $I(\mathbf{i}')$, which was used for checking the acceptability on the right of it, by setting $R(\mathbf{i_{s+1}}(\mathbf{b_1})) := \mathbf{b_2}$ in the considered case.

If $\mathbf{i_{s+1}}(\mathbf{b_2}) := (\mathbf{i_s} + \mathbf{b_2}) \cap \mathbf{y_n} \ne \phi$ and

$$(\mathbf{i_{s+1}}(\mathbf{b_2}) + \mathbf{b_1} + \mathbf{i_t}) \cap \left[\frac{1}{1+r}L', \frac{1}{1-r}L'\right] \ne \phi,$$

then set

$$X'_n := X'_n \cup \{\mathbf{i} = (\mathbf{i_0}, \mathbf{i_1}, \ldots, \mathbf{i_s}, \mathbf{i_{s+1}}(\mathbf{b_2}), \mathbf{i_t}, \ldots, \mathbf{i_{n+1}})\}, n_L(\mathbf{i}) := s + 1$$

and $R(\mathbf{i_{s+1}}(\mathbf{b_2})) := \mathbf{b_1}$.

If $\mathbf{i_{t-1}}(\mathbf{b_1}) := (\mathbf{i_t} + \mathbf{b_1}) \cap \mathbf{y_n} \ne \phi$ and

$$(\mathbf{i_{t-1}}(\mathbf{b_1}) + \mathbf{b_2} + \mathbf{i_s}) \cap \left[\frac{1}{1+r}L', \frac{1}{1-r}L'\right] \ne \phi,$$

then set $X'_n := X'_n \cup \{\mathbf{i} = (\mathbf{i_0}, \mathbf{i_1}, \ldots, \mathbf{i_s}, \mathbf{i_{t-1}}(\mathbf{b_1}), \mathbf{i_t}, \ldots, \mathbf{i_{n+1}})\}$, $n_L(\mathbf{i}) := s$ and $R(\mathbf{i_s}) := \mathbf{b_2}$.

If $\mathbf{i_{t-1}}(\mathbf{b_2}) := (\mathbf{i_t} + \mathbf{b_2}) \cap \mathbf{y_n} \ne \phi$ and

$$(\mathbf{i_{t-1}}(\mathbf{b_2}) + \mathbf{b_1} + \mathbf{i_s}) \cap \left[\frac{1}{1+r}L', \frac{1}{1-r}L'\right] \ne \phi,$$

then set $X'_n := X'_n \cup \{\mathbf{i} = (\mathbf{i_0}, \mathbf{i_1}, \ldots, \mathbf{i_s}, \mathbf{i_{t-1}}(\mathbf{b_2}), \mathbf{i_t}, \ldots, \mathbf{i_{n+1}})\}$, $n_L(\mathbf{i}) := s$, and $R(\mathbf{i_s}) := \mathbf{b_1}$.

Step 4. For each $\mathbf{i} = (\mathbf{i_0}, \mathbf{i_1}, \ldots, \mathbf{i_n}, \mathbf{i_{n+1}}) \in X'_n$, $\mathbf{i_0} = \mathbf{i_{n+1}} = [0, 0]$, perform the following computations: Let $[c_j, d_j] := \mathbf{i_j}$, $j = 1, \ldots, n$, $s := n_L(\mathbf{i})$, $[v_j, w_j] := L(\mathbf{i_j})$, $j = 1, \ldots, s$, $[v_j, w_j] := R(\mathbf{i_j})$, $j = s+1, \ldots, n$, and $[v_{n+1}, w_{n+1}] := R(\mathbf{i_s})$. Construct an increasing point sequence $p(\mathbf{i}) = (p_0, p_1, \ldots, p_{n+1})$, where $p_0 = 0$, which is an approximate solution to SPDP with measurement errors. Introduce positive integer variables $x_1, \ldots, x_{n+1}$. Points p_j, $j = 1, \ldots, n+1$, are determined from $p_j - p_{j-1} = x_j$, $j = 1, \ldots, s$, $p_{j+1} - p_j = x_j$, $j = s+1, \ldots, n$, and $x_{n+1} = p_{s+1} - p_s$. Variables x_j, $j = 1, \ldots, n+1$, represent a solution to the following system of inequalities:

$$v_j \le x_j \le w_j,\ j = 1, \ldots, n+1,$$

$$c_j \le \sum_{h=1}^{j} x_h \le d_j,\ j = 1, \ldots, s,$$

$$c_j \le \sum_{h=j}^{n} x_h \le d_j,\ j = s+1, \ldots, n,$$

$$\frac{1}{1+r}L' \le \sum_{h=1}^{n+1} x_h \le \frac{1}{1-r}L'.$$

There are standard integer programming algorithms to solve the above system of inequalities.

Output set $\{p(\mathbf{i}) \mid \mathbf{i} \in X'_n\}$, which contains a $(\frac{2r}{1-r}, p^*)$-approximate solution for each $p^* \in P^*$.

Similarly to the above adaptation of algorithm ENUM, our dynamic programming algorithm DP can be used for finding an approximate solution of the general case of the error-free SPDP and the problem with measurement errors. The idea is to round input data of the problem so that the number of distinct rounded interpoint distances is sufficiently small to apply a modification of algorithm DP efficiently. More specifically, $n+1$ interpoint distances from the multiset B can be partitioned into several groups, say, q groups, so that a relative or absolute deviation between distances in the same group does not exceed a given value. Then, we can reset distances in the same group to be equal to the average distance in this group and, as a

TABLE 1
Running Time Comparison of ENUM and DP
(Random Data, $n = 1,000$)

q	Running time, msec				Max.number of solutions
	ENUM		DP		
	Average	Std deviation	Average	Std deviation	
40	106	43.1	75	24.8	8
50	111	28.1	85.2	18.5	2
60	125	66	97.8	43.8	4
70	114	41.8	96.3	32.9	4
80	140	42.7	126	38.7	8
90	151	137	140	106	8
100	162	89.6	161	87.3	8
110	154	114	161	108	8
120	175	92.4	196	103	8
130	152	144	193	168	4
140	190	226	241	272	8
150	174	115	282	208	8

result, obtain the number of distinct interpoint distances equal to the number of the groups q. Algorithm DP can be modified for interval computations as follows: While considering $o_j \in A$, an assignment of a rounded interpoint distance $b \in B$ to the left part of the molecule is accepted if the sum of the rounded interpoint distances assigned to the left part so far deviates from o_j within a specified range. A similar rule can be applied for the assignment of a rounded interpoint distance to the right part of the molecule. A detailed implementation of this approach is not straightforward and represents a direction for future research.

6 Experiments

In this section, we present the results of computer experiments with algorithms ENUM, DP, Interval-ENUM, and their comparison with the results of Skiena and Sundaram [26] for PDP and with the results of Blazewicz et al. [2] for SPDP. We denote the algorithm of Skiena and Sundaram [26] as Pyramid-PDP and the algorithm of Blazewicz et al. [2] as First-SPDP. Algorithms ENUM, Interval-ENUM, and DP were implemented in Borland C++ Builder 6, and the tests were run on a portable PC with an Intel Pentium M 2 GHz processor and 480 Mbytes of RAM under Windows XP. Large-scale instances with bigger memory requirements were run on a single processor of a Sun SunFire 6800 supercomputer. All random numbers in our experiments were generated by using uniform distribution. Tests of the algorithm Pyramid-PDP were run on a Sun Sparcstation 2 and those of the algorithm First-SPDP on a PC with Celeron 420 MHz processor and 64 Mbytes of RAM; see [26] and [2].

6.1 Error-Free Data

In the first set of experiments, algorithms ENUM and DP were run for fixed $n = 1,000$ and various values of q. Given q, we randomly generated distinct interpoint distances $u_i \in (100, 2{,}000)$, $i = 1, \ldots, q$. Let k_i denote the number of interpoint distances equal to u_i, $i = 1, \ldots, q$. We randomly generated these numbers such that $\sum_{i=1}^{q} k_i = n + 1$. With numbers u_i and k_i, $i = 1, \ldots, q$, the multiset B is fully determined. We assumed that $o_j = \sum_{i=1}^{j} b_i$ represents the distance between point 0 and a point corresponding to restriction site j in a DNA chain. The multiset A was generated accordingly. Given $n = 1{,}000$, algorithms ENUM and DP were run on 100 instances for every value of q. Corresponding values of average runtimes of the algorithms (column "Average"), standard deviation from the average runtime (column "Std deviation"), and the maximum number of solutions found by algorithm ENUM are given in Table 1.

In the first set of experiments, we observed that algorithm DP outperforms algorithm ENUM if the number, q, of distinct interpoint distances is within 10 percent of the total number $n + 1$ of interpoint distances.

Our second set of experiments was performed using real data about DNA chains taken from the nucleotide database GenBank; see [14]. Given a sequence of nucleotides (an entry in GenBank) and a restriction enzyme, the multisets A and B were determined as if we performed an ideal biochemical experiment. Table 2 presents the results of our second set of experiments.

TABLE 2
Running Time Comparison of ENUM and DP (Real Data)

Entry in GenBank	Restriction enzyme	n	q	Running time, msec		Number of solutions
				ENUM	DP	
D26561	*Alu*I	36	35	< 10	< 10	1
J00277	*Hha*I	38	31	< 10	10	1
J00277	*Hae*III	99	63	50	241	2
NC_006852	*Hla*III	128	110	< 10	10	1
DQ084247	*Hla*III	156	134	20	51	4
NC_005045	*Hla*III	207	148	10	41	1
AM084415	*Alu*I	215	158	78	402	2

Table 2 does not contain information about earlier techniques because, for the PDP, no experimental results over real data were reported in the literature and, for SPDP, only a few such results were reported in [2] with regard to the algorithm First-SPDP. However, the runtime values of First-SPDP were presented in the form "< 1 sec." This information cannot be used for comparison with our algorithms because both algorithms ENUM and DP solve the same real data instances in less than one second too.

Our third set of experiments for large n was run on a single 400 MHz IP35 processor of a Sun SunFire 6800 supercomputer, with 20 Gbytes of memory limit. In this set of experiments, we considered $q = 2$, interpoint distances, $u_1 = 3$ and $u_2 = 5$, and various values of n. The multisets A and B were determined in the same way as in the first set of experiments. Given $q = 2$, $u_1 = 3$, and $u_2 = 5$, algorithms ENUM and DP were run on 100 instances for every $n \in \{100, 150, 200, 250\}$. Moreover, algorithm DP was run on 100 instances for every $n \in \{2000,\ 4000,\ 6000,\ 8000,\ 10000,\ 12000\}$. Algorithm ENUM was not used for these values of n due to the excess of the 20 Gbyte of memory limit. Corresponding values of average runtimes of the algorithms, standard deviation from the average runtime, and the maximum number of solutions are given in Table 3.

We use $f(n)$ to denote the average runtime of algorithm DP as a function of n in the third set of experiments. From Table 3, we observe that, for $n = 1{,}000$,

TABLE 3
Running Time Comparison of ENUM and DP (Random Data, $q = 2$, $u_1 = 3$, and $u_2 = 5$)

n	Running time, sec				Max.number of solutions
	ENUM		DP		
	Average	Std deviation	Average	Std deviation	
100	0.016	0.019	< 0.01	< 0.01	2^8
150	0.093	0.093	< 0.01	< 0.01	2^{11}
200	6.39	24.1	0.05	0.19	2^{18}
250	33.1	84.3	0.26	0.98	2^{20}
2000	–	–	1.38	0.23	–
4000	–	–	7	0.87	–
6000	–	–	19.5	2.07	–
8000	–	–	42.3	4.17	–
10000	–	–	75.2	5.67	–
12000	–	–	120	8.35	–

$f(2n) \leq 2^{2.5} f(n)$, $f(3n) \leq 3^{2.5} f(n)$, $f(4n) \leq 4^{2.5} f(n)$, $f(5n) \leq 5^{2.5} f(n)$, and $f(6n) \leq 6^{2.5} f(n)$. If we assume that $f(n) \leq Cn^{2.5}$ for $n = 2,000$ and some constant C, then the behavior of the function $f(n)$ for $n \in \{4{,}000,\ 6{,}000,\ 8{,}000,\ 10{,}000)\}$ is in accordance with the above inequality. Notice that the theoretical upper bound on $f(n)$ is $O(n^4)$ in this case.

Our fourth set of experiments was performed over the same random data as proposed in [2] and [26]. Algorithms ENUM and DP were run on 100 instances for different values of n. We compared runtimes of the algorithms ENUM, DP, Pyramid-PDP [26], and First-SPDP [2] for each n, see Table 4. The left value of the given time interval denotes the shortest runtime and the right value denotes the longest runtime. We performed additional tests for $n = 30$ and $n = 100$ to demonstrate that ENUM and DP perform well for larger instances.

Our experiments demonstrated that algorithms ENUM and DP are able to solve instances of SPDP with hundreds of restriction sites in less than one second on a standard PC. Algorithm DP outperformed algorithm ENUM on random data if $q \leq 0.1n$. Based on the results of the experiments, the average runtime of algorithm DP is expected to be $O(n^{1.25q})$, whereas its theoretical worst-case runtime is $O(n^{2q})$.

We analyzed real data from the nucleotide database GenBank [14] with regard to the number of distinct interpoint distances q. In this experiment, we cut 43 DNA molecules containing 2,000-200,000 base pairs by 168 enzymes. To reduce search time, we selected only those distinct pairs (DNA molecule, restriction enzyme) which gave instances of SPDP with $n < 300$. As a result, we obtained 3,710 distinct combinations (DNA molecule, restriction enzyme). Among them, the average value of q was equal to $0,935n$, and the standard deviation from the average value was equal to $0,034n$. We also conducted computer experiments with the same 3,710 real instances of SPDP to establish the number of noncongruent solutions. It was equal to 1, 2, and 4 for 3,641, 64, and 5 instances, respectively, and it was never equal to 3 or exceeded 4.

Notice that any solution from the set of noncongruent solutions can correspond to the original DNA and, within the considered model, there is no instrument to determine the closeness of a given solution to the original DNA. If the problem is to verify whether the map of the target DNA is present in a database, every noncongruent solution can be presented for the verification and the results can be analyzed by an expert.

TABLE 4
Running Time Comparison of Pyramid-PDP, First-SPDP, ENUM, and DP (Random Data from That in [2] and [26])

n	Running time, msec			
	Pyramid-PDP	First-SPDP	ENUM	DP
10	20-30	0.14-0.15	0-0.10	0-1.1
12	20-60	0.17-0.2	0-0.10	0-1.3
14	30-50	0.24-0.6	0-0.15	0-1.9
16	30-60	0.31-0.39	0-0.18	0-2.9
18	50-80	0.47-0.52	0-0.21	0-2.4
20	50-80	0.48-0.72	0-0.24	0-5.1
30	–	–	0-1.00	0-20
100	–	–	0-2.50	0-40

The number of noncongruent solutions of SPDP is an important characteristic of the simplified partial digest method, which shows whether the map of the original DNA can be uniquely determined by solving the corresponding instance of SPDP. If there is more than one solution, it is an indication that the used enzyme provides fragment lengths that can be combined to form several distinct DNA maps. In this case, another enzyme can be used to identify the target DNA through solving SPDP.

6.2 Data with Measurement Errors

In the first set of experiments, for the case of measurement errors, we used real data obtained after cutting bacteriophage λ with enzyme $Hind$III; see [20]. The ideal biochemical experiment provided $n = 7$ restriction sites, multiset $B = \{23{,}130,\ 2{,}027,\ 2{,}322,\ 9{,}416,\ 564,\ 125,\ 6{,}557,\ 4{,}361\}$ with interpoint distances listed from the left end to the right end of the molecule and corresponding multiset A. To simulate measurement errors, we used the same simulation method that was used to obtain imprecise input data for the algorithm Pyramid-PDP [26]. That is, we replaced every distance d in the multisets A and B by a random integer number in the interval $[d(1-r), d(1+r)]$, where r is a given relative measurement error. The obtained multisets A' and B' were used as an input for the algorithm Interval-ENUM. The corresponding runtimes of the algorithms and the maximum number of solutions found are given in Table 5.

Recall that the algorithm Interval-ENUM finds one $(\frac{2r}{1-r}, p^*)$-approximate solution for each $p^* \in P^*$. The total number of such approximate solutions can be huge (a product of the lengths of some intervals of integer points), although many of them will be close to one another. It appears that Algorithm Pyramid-PDP can find several approximate solutions for the same $p^* \in P^*$. This observation explains the fact that algorithm Pyramid-PDP sometimes finds more solutions than our algorithm Interval-ENUM.

Our final set of experiments was performed over random data. First, an instance of the error-free SPDP was constructed. Multisets A and B for this instance were

TABLE 5
Experiments with Imprecise Data for Bacteriophage λ and Enzyme $Hind$III, $n = 7$

Relative error r	Running time, msec		Max.number of solutions	
	Pyramid-PDP	Interval-ENUM	Pyramid-PDP	Interval-ENUM
0	40	0.16	1	1
0.005	70	0.88	3	3
0.01	100	0.3	4	4
0.015	970	0.3	4	4
0.02	3320	0.51	8	6
0.025	9120	0.53	8	6
0.03	97370	0.54	16	6
0.04	–	5.5	–	7
0.05	–	4.18	–	11
0.06	–	116.3	–	21

generated according to the probabilistic model in [26]. Perturbed data (with measurement errors) was obtained as in the first set of experiments presented in this section. Table 6 shows the runtimes of algorithms Pyramid-PDP on five random instances and Interval-SPDP on 100 random instances for various combinations of n and r, $10 \leq n \leq 20$ and $0 \leq r \leq 0.02$. The left value of the given time interval denotes the shortest runtime and the right value denotes the longest runtime. As was the case in [26], the number of solutions was not counted in these experiments.

Experiments with imprecise data demonstrate that our algorithm Interval-ENUM is able to reconstruct the linear structure of a DNA molecule with reasonable quality in a short time.

7 Conclusions

We presented an $O(n2^n)$ time ENUM and an $O(n^{2q})$ time dynamic programming algorithm for the error-free case of the SPDP, where n is the number of sites and q is the number of distinct interpoint distances. The algorithms are based on the established combinatorial properties of the problem. We gave examples of the problem with interpoint distances 1 and 2 and all interpoint distances distinct in which there are $2^{\frac{n+2}{3}-1}$ noncongruent solutions. The ENUM was adapted to handle the problem with imprecise input data by providing a set of solutions, which contains a solution with a linear structure that is close to the original DNA. Computer experiments with our algorithms demonstrated that they outperform earlier algorithms (for recovering DNA linear structure) in the runtime while providing the same quality of solution.

Further research on SPDP can be undertaken to adapt our dynamic programming algorithm for finding an approximate solution of the error-free problem and the problem with measurement errors. It would also be interesting to establish a theoretical relationship between the input parameters of the error-free SPDP and the number of its noncongruent solutions. In the general area of the restriction site analysis, an important open question is the computation complexity of the error-free PDP.

TABLE 6
Running Time Comparison of Pyramid-PDP and Interval-ENUM (Perturbed Random Data from That in [26])

n	Running time, sec ($r = 0$)		Running time, sec ($r = 0.005$)	
	Pyramid-PDP	Interval-SPDP	Pyramid-PDP	Interval-SPDP
10	0.02-0.03	0-0.01	0-1.1	0-0.01
12	0.02-0.06	0-0.01	0-4.2	0-0.01
14	0.03-0.05	0-0.01	0-127	0-0.011
16	0.03-0.06	0-0.01	75.9-94.9	0-0.01
18	0.05-0.08	0-0.01	$>$ 5 min	0-0.02
20	0.05-0.08	0-0.01	$>$ 5 min	0.01-0.62
n	Running time, sec ($r = 0.01$)		Running time, sec ($r = 0.02$)	
	Pyramid-PDP	Interval-SPDP	Pyramid-PDP	Interval-SPDP
10	0-7.7	0-0.01	0-152	0-0.9
12	10.6-131	0-0.02	$>$ 5 min	0.1-5.4
14	$>$ 5 min	0-0.71	$>$ 5 min	0.54-96.6
16	$>$ 5 min	0-11.6	$>$ 5 min	$>$ 5 min
18	$>$ 5 min	0.1-63.7	$>$ 5 min	$>$ 5 min
20	$>$ 5 min	$>$ 5 min	$>$ 5 min	$>$ 5 min

Acknowledgments

This work was supported by BIOPTRAIN EU grant, EPSRC Grant GR/S64530/01, KBN Grant 3T11F00227, and INTAS Grant 03-51-5501.

References

[1] J. Blazewicz, P. Formanowicz, and M. Kasprzak, "Selected Combinatorial Problems of Computational Biology," *European J. Operational Research,* vol. 161, pp. 585-597, 2005.
[2] J. Blazewicz, P. Formanowicz, M. Kasprzak, M. Jaroszewski, and W.T. Markiewicz, "Construction of DNA Restriction Maps Based on a Simplified Experiment," *Bioinformatics,* vol. 17, pp. 398-404, 2001.
[3] J. Blazewicz and M. Jaroszewski, "New Algorithm for the Simplified Partial Digest Problem," *Lecture Notes in Bioinformatics,* vol. 2812, pp. 95-110, 2003.
[4] J. Blazewicz and M. Kasprzak, "Combinatorial Optimization in DNA Mapping—A Computational Thread of the Simplified Partial Digest Problem," *RAIRO Operations Research,* vol. 39, pp. 227-241, 2005.
[5] *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Methodologies,* E.K. Burke and G. Kendall, eds. Springer, 2005.
[6] M. Cieliebak and S. Eidenbenz, "Measurement Errors Make the Partial Digest Problem NP-Hard," *Proc. Latin Am. Symp. Theoretical Informatics (LATIN '04),* pp. 379-390, 2004.
[7] M. Cieliebak, S. Eidenbenz, and P. Penna, "Noisy Data Make the Partial Digest Problem NP-Hard," *Lecture Notes in Bioinformatics,* vol. 2812, pp. 111-123, 2003.
[8] K. Danna and D. Nathans, "Specific Cleavage of Simian Virus 40 DNA by Restriction Endonuclease of *Hemophilus influenzae*," *Proc. Nat'l Academy of Sciences USA,* vol. 68, pp. 2913-2917, 1971.
[9] K.J. Danna, G.H. Sack, and D. Nathans, "Studies of Simian Virus 40 DNA. VII. A Cleavage Map of the SV40 Genome," *J. Molecular Biology,* vol. 78, pp. 363-376, 1973.
[10] A. Daurat, Y. Gerard, and M. Nivat, "Some Necessary Clarifications about the Chords' Problem and the Partial Digest Problem," *Theoretical Computer Science,* vol. 347, nos. 1-2, pp. 432-436, 2005.
[11] T.I. Dix and D.H. Kieronska, "Errors between Sites in Restriction Site Mapping," *Computer Applications in the Biosciences,* vol. 4, pp. 117-123, 1988.
[12] K. Dowsland, "Classical Techniques," *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Methodologies,* E.K. Burke and G. Kendall, eds., pp. 19-68, Springer, 2005.

[13] A. Dudez, S. Chaillou, L. Hissler, R. Stentz, M. Champomier-Verges, C. Alpert, and M. Zagorec, "Physical and Genetic Map of the *Lactobacillus sakei* 23K Chromosome," *Microbiology,* vol. 148, pp. 421-431, 2002.
[14] GenBank, http://www.ncbi.nlm.nih.gov/Genbank/index.html, 2007.
[15] Y. Gerard, personal communication, 2006.
[16] J.A. Inglehart and P.C. Nelson, "On the Limitations of Automated Restriction Mapping," *Computer Applications in the Biosciences,* vol. 10, no. 3, pp. 249-261, 1994.
[17] R.B. Kearfott, "Interval Computations: Introduction, Uses, and Resources," *Euromath Bull.,* vol. 2, no. 1, pp. 95-112, 1996.
[18] S. Keis, J.T. Sullivan, and D.T. Jones, "Physical and Genetic Map of the *Clostridium saccharobutylicum* (Formerly *Clostridium acetobutylicum*) NCP 262 Chromosome," *Microbiology,* vol. 147, pp. 1909-1922, 2001.
[19] T. Kuwahara, M.R. Sarker, H. Ugai, S. Akimoto, S.M. Shaheduzzaman, H. Nakayama, T. Miki, and Y. Ohnishi, "Physical and Genetic Map of the *Bacteroides fragilis* YCH46 Chromosome," *FEMS Microbiology Letters,* vol. 207, pp. 193-197, 2002.
[20] T. Maniatis, E.F. Fritsch, and J. Sambrook, *Molecular Cloning, A Laboratory Manual.* Cold Spring Harbor Laboratory, 1982.
[21] M.A. Marra, T.A. Kucaba, N.L. Dietrich, E.D. Green, B. Brownstein, R.K. Wilson, K.M. McDonald, L.W. Hillier, J.D. McPherson, and R.H. Waterston, "High Throughput Fingerprint Analysis of Large-Insert Clones," *Genome Research,* vol. 7, pp. 1072-1084, 1997.
[22] P.A. Pevzner, *Computational Molecular Biology: An Algorithmic Approach.* MIT Press, 2000.
[23] D. Poole, A. Mackworth, and R. Goebel, *Computational Intelligence: A Logical Approach.* Oxford Univ. Press, 1998.
[24] J. Setubal and J. Meidanis, *Introduction to Computational Molecular Biology.* PWS, 1997.
[25] S.S. Skiena, W.D. Smith, and P. Lemke, "Reconstructing Sets from Interpoint Distances," *Proc. Sixth ACM Symp. Computational Geometry,* pp. 332-339, 1990.
[26] S.S. Skiena and G. Sundaram, "A Partial Digest Approach to Restriction Site Mapping," *Bull. Molecular Biology,* vol. 56, pp. 275-294, 1994.
[27] M.S. Waterman, *Introduction to Computational Biology: Maps, Sequences and Genomes.* Chapman and Hall, 1995.
[28] J.D. Watson and F.H.C. Crick, "Genetic Implications of the Structure of Deoxyribonucleic Acid," *Nature,* vol. 171, pp. 964-967, 1953.
[29] L.W. Wright, J.B. Lichter, J. Reinitz, M.A. Shifman, K.K. Kidd, and P.L. Miller, "Computer-Assisted Restriction Mapping: An Integrated Approach to Handling Experimental Uncertainty," *Computer Applications in the Biosciences (CABIOS),* vol. 10, no. 4, pp. 435-442, 1994.

Jacek Blazewicz received the MSc degree in control engineering in 1974, the PhD and Dr. habil. in computer science in 1977 and 1980, respectively, and the dr.h.c. degree from the University of Siegen in 2006. He is a professor of computer science at the Poznan University of Technology, Poland. Currently, he is a deputy director of the Institute of Computing Science and he also holds the position of professor at the Institute of Bioorganic Chemistry of the Polish Academy of Sciences. His research interests include algorithm design and complexity analysis of algorithms, especially in bioinformatics, as well as in scheduling theory. He has published widely in the above fields (more than 290 papers) in many outstanding journals, including the *Journal of Computational Biology*, *Bioinformatics*, *Computer Applications in Biosciences*, *Operations Research*, *IEEE Transactions on Computers*, *IEEE Transactions on Communications*, *Discrete Applied Mathematics*, *Parallel Computing*, *Acta Informatica*, *Performance Evaluation*, *International Transactions of Operations Research*, *European Journal of Operational Research*, *Information Processing Letters*, and *Operations Research Letters*. He is also the author or coauthor of 14 monographs. He is an editor of the International Series of Handbooks in Information Systems (Springer Verlag), as well as a member of the editorial boards of several scientific journals. In 1991, he was awarded the EURO Gold Medal for his scientific achievements in the area of operations research. In 2002, he was elected to be the corresponding member of the Polish Academy of Sciences. He is a senior member of the IEEE.

Edmund K. Burke is the Head of the School of Computer Science at the University of Nottingham. He also leads the Automated Scheduling, Optimisation, and Planning Research Group. He is a member of the Engineering and Physical Sciences Research Council (EPSRC) Peer Review College and is a fellow of the British Computer Society. He is also a member of the General Council of the Operational Research Society and the UK Computing Research Committee (UKCRC). He is the editor-in-chief of the *Journal of Scheduling*, area editor for *Combinatorial Optimisation of the Journal of Heuristics*, associate editor of the *INFORMS Journal on Computing*, and associate editor on the *IEEE Transactions on Evolutionary Computation*. He has also acted as a guest editor of two feature issues of the *European Journal of Operational Research* and is guest editing two forthcoming special issues of the *Annals of Operations Research*. He was a member of the jury to award the European Association of Operational Research Societies (EURO) Gold Medal for 2007. He is the chairman of the steering committee of the international series of conferences on the Practice and Theory of Automated Timetabling (PATAT). He was the cochair of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CISched) held in Hawaii in 2007. He is a director of two spin-off companies, which have built on his research (eventMAP Ltd. and Aptia Solutions Ltd.). He has edited/authored 13 books and has published more than 150 refereed papers. He has also been awarded 41 externally funded projects which have been obtained from a wide variety of sources, including EPSRC, ESRC, BBSRC, EU, Teaching Company Directorate, Joint Information Systems Committee of the UK Higher Education Funding Councils, and commercial organizations.

Marta Kasprzak received the degree in computer science from the Poznan University of Technology, Poland, in 1995. In 2004, she defended her habilitation thesis at the same university within the Faculty of Computing Science and Management. She has been continuously employed at the Institute of Computing Science, Poznan University of Technology since 1994. She focuses her scientific research on bioinformatics/computational biology, mainly on DNA sequencing and DNA mapping with the stress put on theoretical analysis of these problems (computational complexity and modeling with the use of a graph theoretical approach). She received an award from the Prime Minister of Poland for her PhD thesis.

Alexandr Kovalev received the master's degree from the Belorusian State University of Informatics and Radioelectronics in 2005. He is a PhD student at the Institute of Computing Science, Poznan University of Technology, Poland. His current scientific interests include bioinformatics and computer science.

Mikhail Y. Kovalyov received the candidate of sciences degree and the doctor of sciences degree in mathematical cybernetics in 1986 and 1999, respectively. He obtained the professor title in computer science in 2004. He is a professor of the Faculty of Economics at Belarusian State University, Minsk, Belarus. His research interests include combinatorial optimization, scheduling, and logistics. He has published more than 70 papers in international journals and one monograph. He is a member of the editorial board of the *European Journal of Operational Research* and *Computers and Operations Research* and is an associate editor for the *Asia-Pacific Journal of Operational Research*.