New Algorithm for the Simplified Partial Digest Problem*

J. Błażewicz[1,2] and M. Jaroszewski[1,2]

[1] Insitute of Computing Science,
Poznań University of Technology,
Piotrowo 3a, 60-965 Poznań, Poland
{Jacek.Blazewicz, Marcin.Jaroszewski}@cs.put.poznan.pl
[2] Institute of Bioorganic Chemistry,
Polish Academy of Sciences,
Noskowskiego 12, 61-704 Poznań, Poland

Abstract. In the paper, the problem of genome mapping is considered. In particular, the restriction site approach is used for this purpose. A new, efficient algorithm for solving the Simplified Partial Digest Problem is presented. The ideal data as well as data with measurement errors can be handled by this algorithm. An extensive computational experiment proved a clear superiority of the presented algorithm over other existing approaches. In addition, a thorough analysis of the Simplified Partial Digest Problem and a discussion of common experimental errors are given.

1 Introduction

Creation of a physical map is one of the basic steps of genome sequencing process. Such a map of a DNA strand contains the information about locations of short, specific subsequences called markers. There are many ways of physical map construction. One of them is based upon splitting a target strand into many shorter ones, called clones, that overlap each other. Next, each clone is subject to hybridization with a set of short DNA fragments, called probes, unique within the target DNA. The information upon which the original ordering of clones is reconstructed is the set of probes that bind to each clone. Methods and algorithms based on foregoing approach are presented among others in [1,13].

Another way to construct physical maps consists of a digestion of a DNA molecule with restriction enzymes. These enzymes cut DNA strands within short, specific patterns called restriction sites. After digestion, the lengths of obtained fragments are measured and serve as the basic information in the process of a reconstruction of the original ordering. In practice, several variants of this approach are used. Two of the best known are: *the double digest* and *the partial digest.*

* The research has been partially supported by a KBN grant.

In the former problem two restriction enzymes are used. A target DNA is amplified, perhaps using a PCR reaction, and the copies are divided into three sets. Molecules from the first set are digested by one enzyme, molecules from the second set are digested by the other enzyme and molecules from the third set are cut by both enzymes. All digestions are complete for the time span of each reaction is long enough to allow the enzyme to cut the target strand at each occurrence of the restriction site. As the result one obtains three collections of short DNA fragments that correspond to three digestion processes. The lengths of these fragments are measured during a gel electrophoresis process and recorded as three multisets. On the basis of this data locations of restriction sites in the target DNA are reconstructed. Unfortunately, from the combinatorial point of view this is a hard problem and, in addition, a number of equivalent solutions grows exponentially with the length of a strand being mapped [2,6,8,9,10,12,16].

A good alternative is, thus, *the partial digest method* where only one enzyme is used [11,14,15]. After amplification, a target DNA is divided into three sets. Molecules from each set are digested by the same enzyme, but the time span allowed for digestion differs among sets. The reaction times are chosen in such a way that in one of the sets most DNA strands were cut exactly once and in another set, exactly twice. For the third set, the reaction time span must be sufficient to let the enzyme cut all molecules in all ocurrences of the restriction site. As the result one gets three collections of restriction fragments. Again, the most important information are lengths of restriction fragments measured during a gel electrophoresis process. The restriction mapping problem based upon above-mentioned biochemical experiment is known as PDP (the Partial Digest Problem). Efficient backtracking algorithm for solving PDP that fills out the matrix of distances between restriction sites was designed by S. Skiena and co-workers [14]. While the algorithm is known to have an exponential complexity in the worst case, on average, however, it performs quite well. In addition, a modification of the above algorithm was proposed by S. Skiena and G. Sundaram [15] that yields very promising results in the presence of measurement errors. The computational complexity of PDP assuming error-free input data is an open question [13]. M. Cieliebak and co-workers proved the NP-hardness of PDP in the presence of unbounded errors [5].

In this paper, we propose an improved algorithm for solving *the simplified partial digest problem (SPDP)*, introduced in [3], where only two digestion processes are performed. In what follows, we will compare the new algorithm to the one discussed in [3] and to the backtracking algorithm designed by S. Skiena and G. Sundaram [15]. The comparison will include two cases. The first one deals with ideal data, involving no experimental errors, the second deals with data containing measurement errors. As an error model we assume the one discussed in [15] to unify the presentation and to enable comparison with the best approach to PDP known so far.

An organization of the paper is as follows. In Section 2, a description of biochemical experiment and a mathematical formulation of SPDP are given. In Section 3, the algorithm for ideal data case is presented along with a discussion of common experimental errors. Considering length measurement errors to be the most important ones, we adopt the model of errors similar to the one presented in [15]. In Section 4, computational results are given and the comparison with the results obtained in [3] and [15] is done.

2 The Problem and Its Basic Features

In the Simplified Partial Digest Problem (SPDP), similarly as in the Partial Digest Problem, only one restriction enzyme is used [3]. However, in case of SPDP only two digestions are performed. We will call them, respectively, a short digestion and a complete digestion. After amplification, the copies of a target strand are split into two sets. The goal of a short digestion is to have almost all molecules from one of the sets cut in at most one ocurrence of the restriction site. This is assured by properly chosen time span of the reaction. Molecules from the other set are cut in all ocurrences of the restriction site due to the long reaction time span (a complete digestion). Then, as in other methods, the lengths of restriction fragments obtained are measured during a gel electrophoresis process.

Let us now define the problem more formally. Let $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_{2n}\}$ be a multiset of fragment lengths (excluding the length of a whole DNA strand) that are obtained out of a short digestion and let $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_{n+1}\}$ be a multiset of fragment lengths obtained out of a complete digestion, where n denotes a number of restriction sites in the target DNA. Furthermore, let us sort the elements of multiset Γ in non-decreasing order. In this way we obtain list $A = \langle a_1, a_2, \ldots, a_{2n} \rangle$. It is easy to observe that in the ideal case (assuming no experimental errors) $a_i + a_{2n-i+1} = l$, $i = 1, \ldots, n$, where l denotes a length of the target DNA strand. The restriction fragments whose lengths are equal to, respectively, a_i and a_{2n-i+1} will be called *complementary* and a pair of complementary fragments will be denoted by $\{a_i, a_{2n-i+1}\}$, $i = 1, \ldots, n$. Obviously, each pair corresponds to exactly one restriction site in the target molecule. Unfortunately, the real ordering of complementary fragments within a pair is not known as the actual information gets lost during digestion processes. Thus, let $P_i = \langle a_i, a_{2n-i+1} \rangle$ and $P_{2n-i+1} = \langle a_{2n-i+1}, a_i \rangle$ denote permutations of pair $\{a_i, a_{2n-i+1}\}$, $i = 1, \ldots, n$. Next let us label restriction sites as $r_1, \ldots, r_n$ in such a way that condition: $s \leq t \Rightarrow a_s \leq a_t$, $s, t = 1, \ldots, n$ would hold for any restriction sites r_s and r_t. Additionally, let us arbitrarily label one of the ends of the target molecule as r_0, the other as r_{n+1} and let us call them, respectively, the left end and the right end of the map. In what follows, by the notion of *a labeled site* we will understand any restriction site or any one of the ends of the map. Let $\Sigma = \{a_1, \ldots, a_n, a_i - a_1, \ldots, a_i - a_{i-1}, l - a_n, l - a_n - a_1, \ldots, l - a_n - a_{n-1}\}$, $i = 2, \ldots, n$ denote a set of all distances between any two restriction sites r_s and r_t, $s = 1, \ldots, n-1$, $t = s+1, \ldots, n$ and between any restriction site r_s and the nearest end of the map (be it r_0 or r_{n+1}); only for r_n two distances: between r_0 and r_n and between r_n and r_{n+1}, respectively, are included. Some elements of Σ may be identical with regard to the value but still represent distances between different restriction sites and thus Σ is a set. Furthermore, let

$\Phi = \{\phi_1, \phi_2, \ldots, \phi_{2^n}\}$, $\phi_k = \langle c_{k1}, c_{k2}, \ldots, c_{k(n+1)} \rangle$, $c_{ki} \in \Sigma$, $i = 1, \ldots, n+1$, $k = 1, \ldots, 2^n$, denote a set of all *feasible orderings* of n restriction sites. A feasible ordering is defined as the result of *a composition* of n permutations, each permutation being obtained out of a different complementary pair. A composition of two permutations $P_i = \langle a_i, a_{2n-i+1} \rangle$ and $P_j = \langle a_j, a_{2n-j+1} \rangle$ is defined as triple $\langle a_i, a_j - a_i, a_{2n-j+1} \rangle$ if $a_i < a_j$ or triple $\langle a_j, a_i - a_j, a_{2n-i+1} \rangle$ whenever $a_j < a_i$, $i = 1, \ldots, 2n, j = 1, \ldots, 2n$, $i \neq j$. Using the foregoing definition, one can easily extend the notion of the composition to the case where i permutations are involved, $i = 3, \ldots, n$. In addition, one should note that values c_{ki} and $c_{k(i+1)}$, $i = 1, \ldots, n$, $k = 1, \ldots, 2^n$ are correlated. As the foregoing dependency is essential to the problem formulation, we will examine it a little bit further. Let us consider elements c_{ki} and $c_{k(i+1)}$ of ϕ_k, $i = 2, \ldots, n-1$, $k = 1, \ldots, 2^n$. The values of elements c_{ki} and $c_{k(i+1)}$ are determined by locations of restriction sites we will denote by r_s, r_t and r_t, r_u. Let $\{a_s, a_{2n-s+1}\}$, $\{a_t, a_{2n-t+1}\}$ and $\{a_u, a_{2n-u+1}\}$ denote complementary pairs that correspond, respectively, to restriction sites r_s, r_t and r_u. There are four different cases we will consider:

1. Restriction sites r_s, r_t and r_u are located in the left half of the map (the half that begins with r_0). In this case c_{ki} equals $a_t - a_s$ and $c_{k(i+1)}$ equals $a_u - a_t$.
2. Restriction sites r_s, r_t and r_u are located in the right half of the map (the half that ends with $r_{(n+1)}$). In this case we obtain $c_{ki} = a_s - a_t$ and $c_{k(i+1)} = a_t - a_u$.
3. Restriction sites r_s, r_t are located in the left half of the map, while restriction site r_u is located in the right half of the map. In this case $c_{ki} = a_t - a_s$ and $c_{k(i+1)} = l - a_u - a_t$.
4. Restriction site r_s is located in the left half of the map, while restriction sites r_t, r_u are located in the right half of the map. In the mentioned case c_{ki} equals $l - a_t - a_s$ and $c_{k(i+1)}$ equals $a_t - a_u$.

One can also construct similar rules for $i = 1$ as well as $i = n$.

Clearly, Φ contains all feasible as well as unacceptable solutions of SPDP. At the end, let us sort the elements of multiset Λ in non-decreasing order to obtain list $B = \langle b_1, b_2, \ldots, b_{n+1} \rangle$.

Using the above denotations, *the Simplified Partial Digest Problem* (the search version) can be formulated as follows:

Find element ϕ of Φ that satisfies the following criterion:

The list of elements of ϕ, sorted non-decreasingly, is identical to list B.

The above notions are illustrated in Fig. 1.

One can prove that a number of different feasible solutions of the Simplified Partial Digest Problem grows exponentially with a number of restriction sites for some instances.

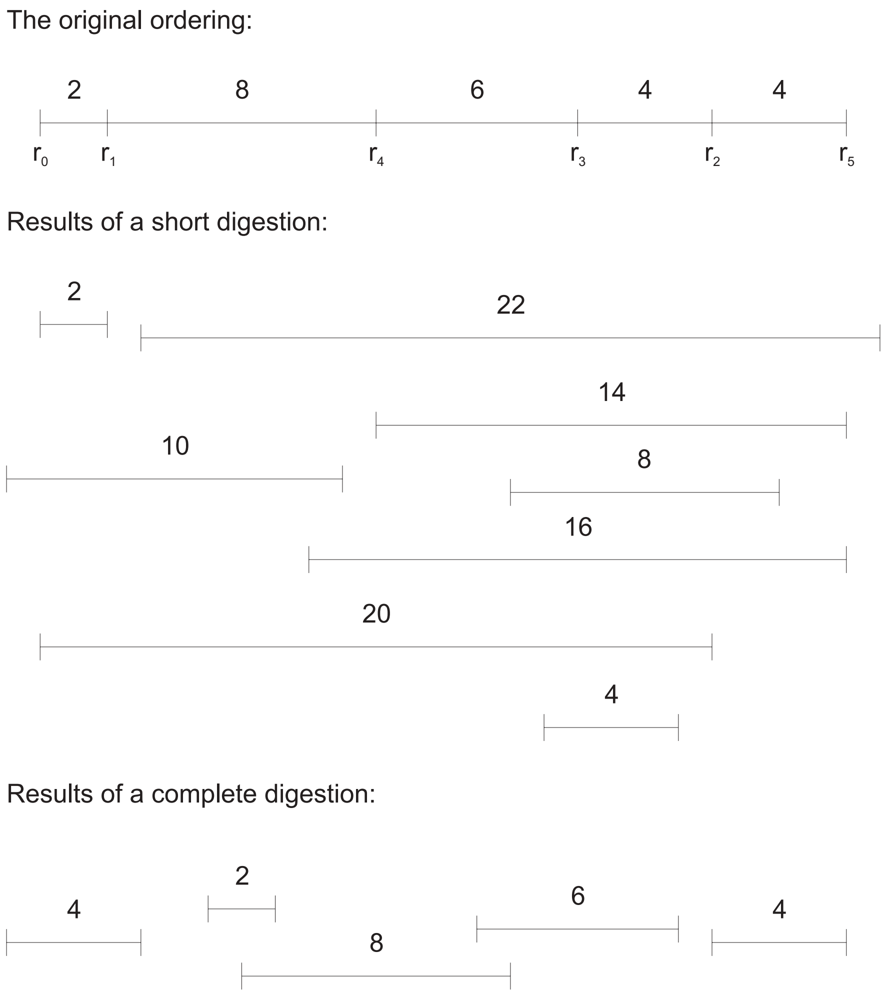
The original ordering:
2
8
6
4
4
r0
r1
r4
r3
r2
r5
Results of a short digestion:
2
22
14
10
8
16
20
4
Results of a complete digestion:
4
2
8
6
4

Fig. 1. An exemplary instance of the Simplified Partial Digest Problem with number of restriction sites n equal to 4 and length l of a target molecule equal to 24. $\Gamma = \{2, 22, 10, 8, 20, 4, 16, 14\}$, thus $A = \langle 2, 4, 8, 10, 14, 16, 20, 22\rangle$, while $\Lambda = \{2, 6, 4, 8, 4\}$ and hence $B = \langle 2, 4, 4, 6, 8\rangle$. There are four complementary pairs: $\{2, 22\}$, $\{4, 20\}$, $\{8, 16\}$, $\{10, 14\}$, each pair having exactly two permutations: $P_1 = \langle 2, 22\rangle$, $P_2 = \langle 4, 20\rangle$, $P_3 = \langle 8, 16\rangle$, $P_4 = \langle 10, 14\rangle$, $P_5 = \langle 14, 10\rangle$, $P_6 = \langle 16, 8\rangle$, $P_7 = \langle 20, 4\rangle$, $P_8 = \langle 22, 2\rangle$. Set of distances $\Sigma = \{2, 4, 8, 10, 14, 2, 6, 8, 12, 4, 6, 10, 2, 6\}$. Set Φ contains $2^4 = 16$ elements, with $\phi_1 = \langle 2, 2, 4, 2, 14\rangle$, $\phi_2 = \langle 2, 2, 4, 6, 10\rangle$, $\phi_3 = \langle 2, 2, 6, 6, 8\rangle$, $\phi_4 = \langle 2, 2, 10, 2, 8\rangle$, $\phi_5 = \langle 2, 6, 2, 10, 4\rangle$, $\phi_6 = \langle 2, 8, 6, 4, 4\rangle$, $\phi_7 = \langle 2, 6, 6, 6, 4\rangle$, $\phi_8 = \langle 2, 12, 2, 4, 4\rangle$ and with elements from ϕ_9 to ϕ_{16} inversely ordered with regard to elements, respectively, from ϕ_1 to ϕ_8. Only two solutions to the problem exist: ϕ_6 and ϕ_{14}, the former one being a composition of permutations: P_1, P_4, P_6, P_7 and the later one being a composition of permutations:P_8, P_5, P_3, and P_2

3 New Algorithm for the Simplified Partial Digest Problem

As we already said in Introduction, the complexity status of the Simplified Partial Digest Problem is open. In [3], an exact algorithm for SPDP was proposed and in the worst case its running time was exponential in the input size. On average, however, this time was much lower. In this paper, new algorithm for SPDP is proposed, which compares favorably with the previous one [3] and with the algorithm proposed for PDP [15].

3.1 Neighbourhood Rules

A construction of neighbourhood rules for each restriction site is an essential step in the process of the algorithm creation. The position of each restriction site is determined by a selection of a permutation of a pair that corresponds to the site. Firstly, let us consider any restriction site r_i, $i = 1, \ldots, n-1$ and its position in a feasible ordering. Clearly, the following possibilities exist:

1. Restriction site r_i is located in the left half of the map. On its left side it has labeled site r_s, $s \in \{0, \ldots, i-1\}$, while on the right side it neighbours restriction site r_t, $t \in \{i+1, \ldots, n\}$.
2. Restriction site r_i is located in the right half of the map. On its left side it has restriction site r_s, $s \in \{i+1, \ldots, n\}$, while on the right side it neighbours labeled site r_t, $t \in \{n+1, 1, \ldots, i-1\}$.

Now, let us analyze the situation for restriction site r_n:

1. Restriction site r_n is located in the left half or in the middle of the map. On its left side it has labeled site r_s, $s \in \{0, \ldots, n-1\}$, while on the right side it neighbours labeled site r_t, $t = n-1$ if $s < n-1$ and $t \in \{1, \ldots, n-2, n+1\}$ if $s = n-1$.
2. Restriction site r_n is located in the right half of the map. On its left side it has labeled site r_s, while on its right side it neighbours labeled site r_t, $t \in \{1, \ldots, n-1, n+1\}$, $s = n-1$ if $t \neq n-1$ and $s \in \{0, \ldots, n-2\}$ whenever $t = n-1$.

The neighbourhood rules are based upon analyses presented above. To make the rules as brief and consistent as possible, we will use Boolean variables. Let α_{ji} denote a Boolean variable that is equal to one whenever difference $a_j - a_i$ is present in a feasible ordering, and is equal to zero otherwise, $i = 1, \ldots, n-1$, $j = i+1, \ldots, n$. Furthermore, let α_{j0} and $\alpha_{(n+1)i}$ denote Boolean variables that are equal to one whenever, respectively, $a_j - a_0$ and $l - a_n - a_i$ are present in a feasible ordering, $i = 0, \ldots, n-1$, $j = 1, \ldots, n$, $a_0 = 0$. Otherwise, respective variables are equal to zero. Notice that equality: $\alpha_{j0} = 1$ may hold even if restriction site r_j is not a neighbour of r_0. In such a case r_j is located in the right half of the map and neighbours r_{n+1} on the right side. Adding the fact that each restriction site has exactly one neighbour on either side, we can formulate the neighbourhood rules as follows:

1. r_1: $\alpha_{10} \cap (\alpha_{21} \cup \alpha_{31} \cup \ldots \cup \alpha_{n1} \cup \alpha_{(n+1)1}) = 1$. Additionally, $(\alpha_{s1} \cap \alpha_{t1}) = 0$, $s, t = 2, \ldots, n+1$ and $s \neq t$.
2. r_2: $(\alpha_{20} \cup \alpha_{21}) \cap (\alpha_{32} \cup \alpha_{42} \cup \ldots \cup \alpha_{n2} \cup \alpha_{(n+1)2}) = 1$, i.e. $(\alpha_{20} \cup \alpha_{21}) = 1$ and $(\alpha_{32} \cup \alpha_{42} \cup \ldots \cup \alpha_{n2} \cup \alpha_{(n+1)2}) = 1$. Additionally, $(\alpha_{20} \cap \alpha_{21}) = 0$ and $(\alpha_{s2} \cap \alpha_{t2}) = 0$, $s, t = 3, \ldots, n+1$ and $s \neq t$.
3. ...
4. r_k, $k = 3, \ldots, n-2$: $(\alpha_{k0} \cup \alpha_{k1} \cup \alpha_{k2} \cup \ldots \cup \alpha_{k(k-1)}) \cap (\alpha_{(k+1)k} \cup \alpha_{(k+2)k} \cup \ldots \cup \alpha_{nk} \cup \alpha_{(n+1)k}) = 1$,i.e. $(\alpha_{k0} \cup \alpha_{k1} \cup \alpha_{k2} \cup \ldots \cup \alpha_{k(k-1)}) = 1$ and $(\alpha_{(k+1)k} \cup \alpha_{k+2)k} \cup \ldots \cup \alpha_{nk} \cup \alpha_{(n+1)k}) = 1$. Additionally, $(\alpha_{sk} \cap \alpha_{tk}) = 0$, $s, t = k+1, \ldots, n+1, s \neq t$ and $(\alpha_{ks} \cap \alpha_{kt}) = 0$, $s, t = 0, \ldots, k-1$ and $s \neq t$.
5. ...
6. r_{n-1}: $(\alpha_{(n-1)0} \cup \alpha_{(n-1)1} \cup \alpha_{(n-1)2} \cup \ldots \cup \alpha_{(n-1)(n-2)}) \cap (\alpha_{n(n-1)} \cup \alpha_{(n+1)(n-1)}) = 1$, i.e. $(\alpha_{(n-1)0} \cup \alpha_{(n-1)1} \cup \alpha_{(n-1)2} \cup \ldots \cup \alpha_{(n-1)(n-2)}) = 1$ and $(\alpha_{n(n-1)} \cup \alpha_{(n+1)(n-1)}) = 1$. Additionally, $(\alpha_{n(n-1)} \cap \alpha_{(n+1)(n-1)}) = 0$ and $\alpha_{(n-1)s} \cap \alpha_{(n-1)t}$, $s, t = 0, \ldots, n-2$ and $s \neq t$.
7. r_n: $(\alpha_{n0} \cup \alpha_{n1} \cup \alpha_{n2} \cup \ldots \cup \alpha_{n(n-2)} \cup \alpha_{n(n-1)}) \cap (\alpha_{(n+1)0} \cup \alpha_{(n+1)1} \cup \alpha_{(n+1)2} \cup \ldots \cup \alpha_{(n+1)(n-2)} \cup \alpha_{(n+1)(n-1)}) = 1$. Additionally, $(\alpha_{n(n-1)} \cup \alpha_{(n+1)(n-1)}) = 1$, $(\alpha_{ns} \cap \alpha_{nt}) = 0$, $(\alpha_{(n+1)s} \cap \alpha_{(n+1)t}) = 0$ and $(\alpha_{ns} \cap \alpha_{(n+1)s}) = 0$, $s, t = 0, \ldots, n-1$ and $s \neq t$.
8. The additional rule: $(\alpha_{20} \cup \ldots \cup \alpha_{(n+1)0}) = 1$ and $(\alpha_{s0} \cap \alpha_{t0}) = 0$, $s, t = 2, \ldots, n+1$ and $s \neq t$.

3.2 Construction of Distance Matrix M

For each restriction site r_j, $j = 1, \ldots, n$, the neighbourhood rules contain the product of two sums, each sum representing the set of feasible candidates for the left or the right neighbour of the restriction site. At most one element of each sum equals one for, as it has been pointed out above, each restriction site neighbours exactly one labeled site on the left and exactly one labeled site on the right. Moreover, at least one element of each sum equals one, as each restriction site clearly must have a neighbour on either side. For $j = 1, \ldots, n-1$ let $\Delta_j = \langle \delta_{j0}, \ldots, \delta_{j(j-1)}, 0, \ldots, 0 \rangle$ and $\Omega_j = \langle \omega_{(j+1)j}, \ldots, \omega_{(n+1)j}, 0, \ldots, 0 \rangle$, $\delta_{ji} = a_j - a_i$, $i = 0, \ldots, j-1$, $\omega_{(n+1)j} = l - a_n - a_j$ and $\omega_{ij} = a_i - a_j$, $i = j+1, \ldots, n$, denote the n dimensional vectors of differences that correspond, respectively, to the first and the second sum in the product given for restriction site r_j. Furthermore, let us extend the above denotations to restriction site r_n. Thus, let $\Delta_n = \langle \delta_{n0}, \ldots, \delta_{n(n-1)} \rangle$ and $\Omega_n = \langle \omega_{(n+1)0}, \ldots, \omega_{(n+1)(n-1)} \rangle$, $\delta_{ni} = a_n - a_i$, $i = 0, \ldots, n-1$, $\omega_{(n+1)i} = l - a_n - a_i$, $i = 0, \ldots, n-1$ denote the n dimensional vectors of differences that correspond to the first and the second sum in the product given for restriction site r_n. Moreover, let $\Omega_0 = \langle \omega_{20} \cup \ldots \cup \omega_{(n+1)0} \rangle$, $\omega_{(n+1)0} = l - a_n$ and $\omega_{i0} = a_i$, $i = 2, \ldots, n$ denote the n dimensional vector that corresponds to sum $\alpha_{20} \cup \ldots \cup \alpha_{(n+1)0}$. Square matrix M of distances between labeled sites, based on the neighbourhood rules, is constructed in the following way:

1. The i-th row of matrix M equals Δ_{i+1}, $i = 1, \ldots, n-1$.
2. The n-th row of matrix M equals Ω_n.

One can prove the following properties of matrix M:

1. j-th column of M equals Ω_{j-1}, $j = 1, \ldots, n$.
2. Construction of a feasible solution of SPDP consists in picking up exactly one element in each row and exactly one element in each column of the matrix, however, these elements may not be chosen arbitrarily.

The algorithm to be described in the next subsection is based on the above property 2.

3.3 The Algorithm

The proposed algorithm operates on matrix M of distances between restriction sites in order to eliminate all elements, but one, in each row and each column (see property 2 above), according to the rules that are specified below. In what follows by the notion of a main element we will understand such an element of M that forms any feasible solution of SPDP. One can prove that there is always at least one main element in each row and in each column of the matrix, however, no two main elements in any row or in any column form the same solution (see property 2 above). Due to the ambiguity that arises whenever it is impossible to establish the main element in a row or in a column (to establish means to distinguish the main element, that belongs to a solution constructed, from other non-zero elements in a row or in a column basing upon available knowledge), the algorithm performs random choices that may lead to non-feasible solutions to the Simplified Partial Digest Problem. Thus, the algorithm is recursive and enables backtracking to the stage where a false random choice has been made if a non-feasible solution has been obtained. Firstly, we will introduce some basic notions and definitions, then the stages of the algorithm will be presented.

Let p denote a level of recurrence. At the beginning $p = 1$. Let $f(p)$, $f(p) = 1, \ldots, n-1$ denote an index of the first row for which it is impossible to establish the main element at current level of recurrence p (one can prove that any unprocessed row at a current level of recurrence may be selected for that purpose). Furthermore, let $N_M(value)$ and $N_B(value)$ denote, respectively, a number of occurrences of elements of matrix M and a number of occurrences of elements of list B that are equal to *value*. Additionally, while considering any element m_{ij} of matrix M we will assume that $j = 1, \ldots, i+1$ for $i = 1, \ldots, n-1$ and $j = 1, \ldots, n$ for $i = n$. Let v denote an auxiliary variable used to indicate whether, at current level of recurrence p, a random choice of the main element (in $f(p)$-th row) is necessary.

Now let us present some definitions:

A disposable element - positive element m_{ij} for which $N_B(m_{ij}) = 0$. Thus, a disposable element does not have its counterpart on list B and, as list B contains only the lengths of fragments of the original ordering, a disposable element cannot form any feasible solution.

A solitary element of a column - positive element m_{ij} for which the following holds: $\neg(\exists k : m_{kj} > 0)$, where $k = j-1, \ldots, i-1, i+1, \ldots, n$ if $j \neq 1$ and $k = 1, \ldots, i-1, i+1, \ldots, n$ otherwise. Any solitary element of a column forms a feasible solution of SPDP (see property 2 of the matrix), unless a false random choice has been made before.

A solitary element of a row - positive element m_{ij} for which the following holds: $\neg(\exists k : m_{ik} > 0)$, where $k = 1, \ldots, j-1, j+1, \ldots, i+1$ if $i \neq n$ and $k = 1, \ldots, j-1, j+1, \ldots, n$ otherwise. Any solitary element of a row forms a feasible solution of SPDP (see property 2 of the matrix), unless a false random choice has been made before.

A unique element - positive element m_{ij} for which the following holds: $N_M(m_{ij}) = N_B(m_{ij})$. Any unique element forms a feasible solution of SPDP (see property 2 of the matrix), unless a false random choice has been made before.

A deletion - an operation that may proceed in two separate ways, depending on the kind of the element being deleted: a deletion of a disposable element consists in an assignment of 0 to the corresponding entry of the matrix, while a deletion of any other element of M or element of B consists in an assignment of $-p$ to the proper entry. These assignments allow for easy backtracking in the case a non-feasible solution of SPDP were constructed.

A consistency check - an operation performed on positive element m_{ij}, just deleted, which consists in comparison of both $N_M(m_{ij})$ and $N_B(m_{ij})$. The result of a check is negative whenever $N_M(m_{ij}) < N_B(m_{ij})$, since in such a case there are too few elements of the matrix that equal m_{ij} to construct any acceptable solution of SPDP.

A labeling - an operation performed on main element m_{ij} of a constructed solution. A labeling consists in an assignment of $-p$ to the proper entry of the matrix and in a deletion of an element of list B that equals m_{ij}. The deletion is performed in order to prevent a usage of elements of the matrix that equal m_{ij} too many times in the constructed solution.

Now, we may give a description of Algorithm 1 that finds a feasible solution of SPDP. On Fig. 2, a high-level description of the algorithm is given, while performed steps are described below in a greater detail.

Algorithm 1

1. Delete all disposable elements from matrix M.
2. Label m_{ij}. Delete all positive elements, but m_{ij}, in i-th row and j-th column of the matrix. After a deletion of each element perform the consistency check for this element. Mark i-th row and j-th column as processed. If the result of any check is negative, a feasible solution cannot be constructed anymore, thus, proceed to step 7.

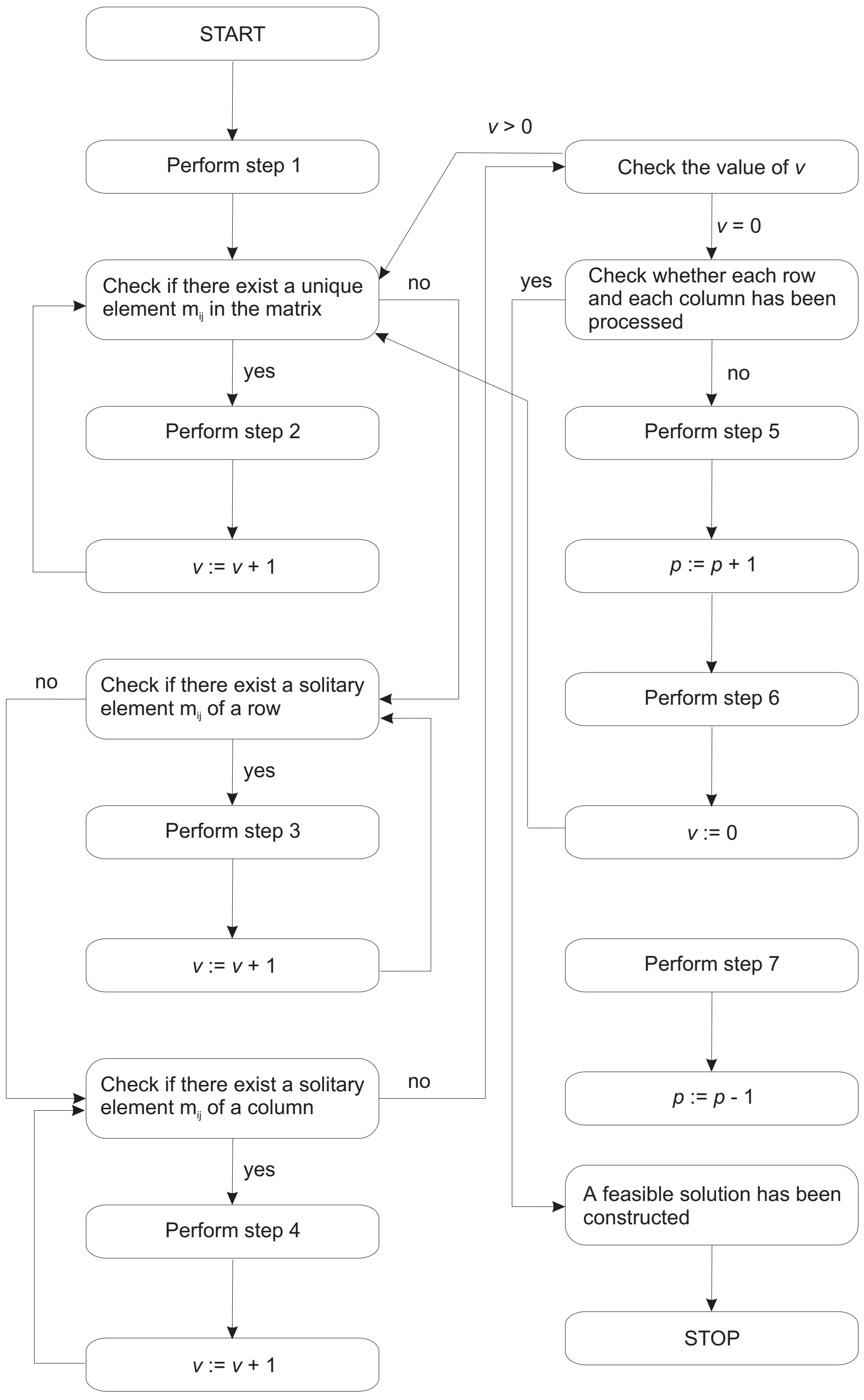
START
Perform step 1
Check if there exist a unique element m_{ij} in the matrix
no
yes
Perform step 2
$v := v + 1$
Check if there exist a solitary element m_{ij} of a row
no
yes
Perform step 3
$v := v + 1$
Check if there exist a solitary element m_{ij} of a column
no
yes
Perform step 4
$v := v + 1$
$v > 0$
Check the value of v
$v = 0$
Check whether each row and each column has been processed
yes
no
Perform step 5
$p := p + 1$
Perform step 6
$v := 0$
Perform step 7
$p := p - 1$
A feasible solution has been constructed
STOP

Fig. 2. A high-level description of the algorithm

3. Label m_{ij}. Delete all positive elements, but m_{ij}, in j-th column of the matrix. After a deletion of each element perform the consistency check for this element. Mark i-th row and j-th column as processed. If the result of any check is negative, a feasible solution cannot be constructed anymore, thus, proceed to step 7.
4. Label m_{ij}. Delete all positive elements, but m_{ij}, in i-th row of the matrix. After a deletion of each element perform the consistency check for this element. Mark i-th row and j-th column as processed. If the result of any check is negative, a feasible solution cannot be constructed anymore, thus, proceed to step 7.
5. Choose the first, not selected previously, positive element $m_{f(p)j}$, where $j = 1, \ldots, f(p) + 1$, in $f(p)$-th row of the matrix. If no such element is found, a feasible solution cannot be constructed anymore, thus, proceed to step 7.
6. Label $m_{f(p)j}$. Delete all positive elements, but $m_{f(p)j}$, in $f(p)$-th row and j-th column of the matrix. After a deletion of each element perform the consistency check for this element. Mark $f(p)$-th row and j-th column as processed. If the result of any check is negative, a feasible solution cannot be constructed anymore, thus, proceed to step 7.
7. Backtrack to the stage of the last random choice undoing all the changes made during current level of recurrence p. They are fairly easy to track as each time an element of M or B is deleted, value $-p$ is assigned to the corresponding entry.

The above algorithm finds only the first feasible solution of the considered problem, but it can be modified to find all of them as it has been done for the needs of testing. Due to the ambiguities that may arise during construction of a solution, the algorithm has an exponential complexity in the worst case. However, its mean behavior is much better, what has been verified by the series of tests, results of which are shown in Section 4.

Algorithm 1 in its basic form could handle ideal data. However, it can be modified to handle data containing errors. This issue is briefly discussed below. The most common experimental errors are caused by imprecise measurement of the lengths of restriction fragments. According to [4], a 2%-5% relative measurement error is achievable. Thus, we have adopted the same model of measurement errors as given in [15], assuming a relative measurement error to range from 0.5% to 5% of the measured length.

Another type of errors is caused by missing restriction fragments. This can be due to approximately equal lengths of fragments, which cannot be then discriminated on a gel. On the other hand, this type of errors may be caused by the fact that certain sites are less likely to be cut than the others, so some fragments might not occur in sufficient quantity to be measured. However, in our SPDP approach such fragments are rather easily traced because there are only two digestion processes, respectively, with a quite short time span and with a very long one, which result in a relatively low number of restriction fragments as compared to the standard Partial Digest Problem. What is more, resulting fragments of a short digestion must form complementary pairs of the known total length. Adding the fact that the number of restriction sites is known, one can usually reconstruct missing fragments.

Summing up, we assume that measurement errors are the most important ones and for the model of such errors adopted from [15] we propose the algorithm that is very similar to the one for error-free input data. There are only two differences:

1. Instead of numbers, intervals are used that correspond to the lower and upper bound of the length of each fragment obtained out of the biochemical experiment (i. e. for measured length l and relative error r, an interval $\langle l_{min}, l_{max} \rangle$ such that $l(1-r) \leq l_{min} \leq l$ and $l \leq l_{max} \leq l(1+r)$ is assumed).
2. All operations performed by the algorithm for ideal data have been changed to fit the rules of interval arithmetic.

In the next section, extensive computational experiments are described, that characterize favorably the approach proposed.

4 Computational Results

In this section, we present the results of tests of the new algorithm in the case of error-free as well as noisy input data. The algorithm has been implemented in DJGPP C++. The tests were run on the PC with Pentium 670MHz processor and 128MB RAM under Windows 98 system. For the sake of comparison, the results obtained for the algorithm solving PDP, presented in [15], and those obtained for the first algorithm solving SPDP, presented in [3], have been added. Running times, numbers of equivalent solutions and relative error rates (in the case of noisy data) of all algorithms are compared. The number of equivalent solutions can be decreased by applying the grouping percentage factor [15], but it can also lead to the loss of feasible solutions. Since no grouping factor has been implemented for both algorithms solving SPDP, the results presented for PDP correspond to the grouping percentage factor equal to zero. In the tables below A_{PDP} stands for the algorithm presented in [15], A_{SPDP} stands for the algorithm presented in [3]. A_{NEW} denotes our just proposed approach, while *msec* stands for miliseconds.

4.1 Error-Free Input Data

At first, we have compared all three algorithms, i.e. solving PDP and two our SPDP approaches on randomly generated instances. Table 1 presents running times for tested algorithms.
The value of each entry in Table 1 represents an average of ten runs of the algorithm. We see that A_{SPDP} as well as A_{NEW} outperform the other algorithm.

The other series of tests for error-free input data have been performed for A_{NEW} only. The data have been obtained by cutting DNA chains taken from [7] in the sites recognizable by restriction enzymes AluI, HaeIII, HhaI and NlaIII. The lengths of restriction fragments created in such a way are indistinguishable

Table 1. Computational results for randomly generated instances (error-free input data)

	Running time		
n	A_{PDP}[sec]	A_{SPDP}[msec]	A_{NEW}[msec]
10	$0.02 - 0.03$	< 10	< 10
12	$0.02 - 0.06$	< 10	< 10
14	$0.03 - 0.05$	< 10	< 10
16	$0.03 - 0.06$	< 10	< 10
18	$0.05 - 0.08$	< 10	< 10
20	$0.05 - 0.08$	< 10	< 10

from data coming from an ideal biochemical experiment. Table 2 shows the results of computations for 11 instances based on real DNA chains, whose lengths range from $2,7$ kbp to $8,9$ kbp.

Table 2. Computational results of SPDP for instances based on real DNA chains (A_{NEW}, error-free input data)

Accession number in GenBank	Restriction enzyme	Number of restriction sites	Computation time [msec]
L13460F	HaeIII	5	< 10
K00470F	HhaI	8	< 10
K00470F	NlaIII	12	< 10
J00277F	NlaIII	19	< 10
D26561F	HaeIII	21	< 10
K00470F	HaeIII	21	< 10
L13460F	NlaIII	22	< 10
D26561F	NlaIII	33	< 10
D26561F	AluI	36	< 10
J00277F	HhaI	38	< 10
J00277F	HaeIII	99	< 10

4.2 Data with Measurement Errors

As a test instance for erroneous data, the restriction map of bacteriophage λ, see [15], was used, as well as randomly generated instances with even number of restriction sites n ranging from 10 to 20. In all cases length l of any restriction fragment was replaced by an interval $\langle l_{min}, l_{max}\rangle$ such that $l(1-r) \le l_{min} \le l$ and $l \le l_{max} \le l(1+r)$, where r denotes a relative error rate. Table 3 shows the results of computations for the restriction map of λ bacteriophage cut by enzyme HindIII. It resulted in seven restriction sites, the distances between adjacent sites being, respectively, $23130, 2027, 2322, 9416, 564, 125, 6557, 4361$ (cf. [15]). For each instance of the problem, numbers of equivalent solutions and running times are presented. Again, both algorithms for SPDP compare favorably with the other algorithm as far as running times are concerned, while having the same numbers of equivalent solutions.

Table 3. Computational results for erroneous instances based on bacteriophage λ (∞ means that no solution has been found in 5 minutes time span)

	Running time			Number of solutions		
Relative error r	A_{PDP}[sec]	A_{SPDP}[msec]	A_{NEW}[msec]	A_{PDP}	A_{SPDP}	A_{NEW}
1.0%	0.04	< 10	< 10	1	2	1
2.0%	0.53	< 10	< 10	1	2	2
3.0%	5.55	< 10	< 10	2	2	2
4.0%	24.17	< 10	< 10	2	2	2
5.0%	∞	< 10	< 10	–	2	2

Additionally, we have performed similar tests, for A_{NEW} only, on randomly generated instances with number of restriction sites n equal to 10 and relative measurement error rate r ranging from 1% to 5%. The results of the test are shown in Table 4.

Table 4. Computational results for randomly generated erroneous instances (n = 10, A_{NEW})

	Running time[msec]	Number of solutions	
Relative error r		minimal	maximal
1.0%	< 10	1	2
2.0%	< 10	1	2
3.0%	< 10	1	8
4.0%	< 10	3	10
5.0%	< 10	12	48

Next all algorithms have been compared on randomly generated erroneous instances. The results of the tests are reported in Table 5. Ten different problem instances were created for each value of n and r. The left bound of the given time interval denotes the best (shortest) and the right bound the worst (longest) running time.

Table 5. Computational results of SPDP for randomly generated instances (∞ means that no solution has been found in 5 minutes time span

	$r = 0,5$			$r = 1,0$		
n	A_{PDP}[sec]	A_{SPDP}[msec]	A_{NEW}[msec]	A_{PDP}[sec]	A_{SPDP}[msec]	A_{NEW}[msec]
10	$0 - 1,10$	< 10	< 10	$0 - 7,70$	< 10	< 10
12	$0 - 4,42$	< 10	< 10	$10,6 - 131$	< 10	< 10
14	$0 - 127$	< 10	< 10	∞	< 10	< 10
16	$75,9 - 94,9$	< 10	< 10	∞	< 10	< 10
18	∞	< 10	< 10	∞	< 20	< 10
20	∞	< 20	< 10	∞	< 20	< 60
	$r = 1,5$			$r = 2,0$		
n	A_{PDP}[sec]	A_{SPDP}[msec]	A_{NEW}[msec]	A_{PDP}[sec]	A_{SPDP}[msec]	A_{NEW}[msec]
10	$0 - 25,5$	< 10	< 10	$0 - 152$	< 10	< 10
12	∞	< 10	< 10	∞	< 20	< 10
14	∞	< 40	< 10	∞	< 400	< 50
16	∞	< 20	< 10	∞	< 100	< 50
18	∞	< 460	< 60	∞	< 5400	< 60
20	∞	< 12060	< 60	∞	< 83000	< 110

5 Conclusions

In the paper, the Simplified Partial Digest Problem, important for genome mapping, has been considered. The new algorithm for SPDP has been described. Again, computational experiments prove its clear superiority over the other approaches: PDP one [15] and the previous SPDP algorithm [3]. The advantage of the new algorithm is evident for both: error-free data and data with measurement errors.

6 Acknowledgement

The authors are greatly indepted to Michael Waterman for his valuable comments and a discussion of the approach.

References

1. Alizadeh, F., Karp, R.M., Weisser, D.K., Zweig, G.: Physical mapping of chromosomes using unique end-probes. Journal of Computational Biology **2** (1995) 159–184
2. Bellon, B.: Construction of restriction maps. Comput. Appl. Biol. Sci. **4** (1988) 111–115
3. Błażewicz, J., Formanowicz, P., Jaroszewski, M., Kasprzak, M., Markiewicz, W.T.: Construction of DNA restriction maps based on a simplified experiment. Bioinformatics **5** (2001) 398–404
4. Chang, W.I., Marr, T.: Personal communication to S. Skiena (1992)

5. Cieliebak, M., Eidenbenz, S., Penna, P.: Noisy Data Make the Partial Digest Problem NP-hard. Technical Report 381, ETH Zurich, Department of Computer Science (2002)
6. Dix, T.I., Kieronska, D.H.: Errors between sites in restriction site mapping. Comput. Appl. Biol. Sci. **4** (1988) 117–123
7. http://www.ncbi.nlm.nih.gov/GenBank/
8. Goldstein, L., Waterman, M.S.: Mapping DNA by stochastic relaxation. Adv. Appl. Math. **8** (1987) 194–207
9. Grigorjev, A.V., Mironov, A.A.: Mapping DNA by stochastic relaxation: a new approach to fragment sizes. Comput. Appl. Biol. Sci. **6** (1990) 107–111
10. Pevzner, P.A.: Physical mapping and alternating eulerian cycles in colored graphs. Algorithmica **13** (1995) 77–105
11. Rosenblatt, J., Seymour, P.: The structure of homeometric sets. SIAM J. Alg. Disc. Math. **3** (1982) 343–350
12. Schmitt, W., Waterman, M.S.: Multiple solutions of DNA restriction mapping problem. Adv. Appl. Math. **12** (1991) 412–427
13. Setubal, J., Meidanis, J.: Introduction to Computational Biology. PWS Publishing Company, Boston (1997)
14. Skiena, S.S., Smith, W.D., Lemke, P.: Reconstructing sets from interpoint distances. In: Proc. Sixth ACM Symp. Computational Geometry (1990) 332–339
15. Skiena, S.S., Sundaram, G.: A partial digest approach to restriction site mapping. Bulletin of Mathematical Biology **56(2)** (1994) 275–294
16. Waterman, M.S.: Introduction to Computational Biology. Maps, Sequences and Genomes. Chapman & Hall, London (1995)