DNA Sequencing with Positive and Negative Errors

J. Błażewicz[†*], P. Formanowicz[†], M. Kasprzak[†], W. T. Markiewicz[‡], J. Węglarz[†‡]

Abstract

The problem to be addressed in this paper is concerned with DNA sequencing by hybridization. An algorithm is proposed that solves a computational phase of this approach in the presence of both positive and negative errors resulting from the hybridization experiment. No a priori knowledge of the nature and source of these errors is required. An extensive set of computational experiments showed that the algorithm behaves surprisingly well if only positive errors appear. The general case, where positive and negative errors occur, can be also solved satisfactorily for an error rate up to 10%.

[†]Institute of Computing Science, Poznań University of Technology, Piotrowo 3A, 60-965 Poznań, Poland.

[‡]Institute of Bioorganic Chemistry, Polish Academy of Sciences, Noskowskiego 12, 61-704 Poznań, Poland.

[*]To whom correspondence should be directed. E-mail: blazewic@sol.put.poznan.pl; phone: +48 61 8790790; fax: +48 61 8771525.

1 Introduction

One of the most challenging issues arising in molecular biology is to read an unknown DNA sequence. The problem addressed in this paper is DNA sequencing - aiming at discovering the exact sequence of nucleotides in a relatively short DNA segment (100-400 bp long).

One of the sequencing methods is *sequencing by hybridization* (SBH), which has been proposed, e.g. by Drmanac et al. (1989), Khrapko et al. (1989), Southern et al. (1992) and Markiewicz et al. (1994). This paper is concerned with the computational part of the SBH method. It is known for some time (Pevzner, 1989) that in the case of ideal hybridization experiment, an original DNA sequence can be reconstructed in polynomial time. The proposed approach is based on a transformation of the computational part of the problem into the one of finding an Eulerian path. On the other hand, an occurrence of errors in the hybridization experiment results in an NP-hard computational problem (Gallant et al., 1980; Gallant, 1983). Two cases can occur. In case of *negative errors*, i.e. where some data (from which a DNA sequence is to be reconstructed) are missing, several algorithms have been proposed (Bains, 1991; Bains and Smith, 1988; Błażewicz et al., 1997b; Guenoche, 1992; Khrapko et al., 1989; Pevzner, 1989), the one presented in (Błażewicz et al., 1997b) being the most general one. The general case where *positive errors* (some additional data which are not present in an ideal experiment) appeared to be more difficult conceptually, and it received less attention in the literature. The only two approaches have been concerned either with the assembling stage of DNA sequence recognition (Drmanac et al., 1991) or have assumed an additional knowledge about rates of negative and positive error occurrence (Lipshutz, 1993). Moreover, both approaches assumed that only first or the last bit (nucleotide) in the obtained data can be erroneous.

In the present paper a new method for DNA sequencing using SBH is proposed. It is assumed that data are generated by the hybridization experiment and can contain positive and negative errors. No *a priori* knowledge about the nature nor the rate of the errors is required. The method is based on reduction to a variant of a *Selective Traveling Salesman Problem*. An implementation of the method uses some ideas of (Laporte and Martello, 1990), adapted for solving the particular problem in question. An extensive set of computational experiments shows that the method behaves surprisingly well if only positive errors appear. The general case, where positive and negative errors occur can be also solved satisfactorily for an error rate up to 10%.

Let us set up the problem more formally. In general, the hybridization experiment is based on the property of single-stranded nucleic acids to form a complex (antiparallel) with a complementary strand of a nucleic acid (Waterman, 1995; Pevzner and Waterman, 1995; Błażewicz et al., 1996). All short fragments of nucleic acids (oligonucleotides) of length l are used in the hybridization experiment (so-called library of 4^l different oligonucleotides), and thus, the formation of the complex indicates the occurrence of a sequence complementary to the oligonucleotide in the nucleic acid. It is detected by a nuclear or spectroscopic detector. As a result of the experiment one gets a set (called *spectrum*) of all l-long oligonucleotides (l-mers) that are known to hybridize with the DNA sequence N of the known length n (i.e., the l-mers are substrings of string N). In a case of ideal data we have thus $|spectrum| = n - l + 1$. In some cases, however, one can obtain less than $n - l + 1$ fragments, either due to experimental problems or as a result of the structure of sequence N (for example, it can contain repeated subsequences). In this case negative errors occur. On the other hand, there can be also positive errors, i.e. not each fragment (oligonucleotide) present in *spectrum* is a part of the original sequence.

The sequencing of DNA by hybridization (SBH) can be realized by two different approaches described in the literature. In the first approach a labeled nucleic acid is used to probe an array of immobilized oligonucleotides (Southern et al., 1992; Fodor et al., 1991; Khorlin et al., 1991). In the second method the opposite approach is used: immobilized nucleic acids are hybridized with labeled oligonucleotides (Drmanac et al., 1989).

In the first approach one can formulate the experimental conditions in which the whole set of l-long oligonucleotides (a full combinatorial library) is used. Such an experiment can be performed using e.g. oligonucleotide arrays of high density (Lipshutz et al., 1995; Chee et al., 1995; Fodor et al., 1996; Hacia et al., 1996). Due to the differences in the thermodynamic stability of perfectly matched duplexes of the same length l resulting in the first instance from differences in a A/T, G/C content, it is not possible to perform an ideal hybridization experiment in general. The addition of chaotropic salts into the hybridization mixtures can reduce this source of errors to some extent (Maskos and Southern, 1993). However, one might assume that a hybridization experiment can supply a data set that contains a smaller number of erroneous than correct l-mers and thus, the maximum possible number of l-mers from *spectrum* should be used in a reconstruction procedure of sequence N. In the same time, performing a hybridization experiment under less stringent conditions will certainly result in positive errors. Their number will differ with the sequence of N, hybridization conditions, length l of oligonucleotides in a probing oligonucleotide array, and the sequence of each particular l-long oligonucleotide in a probing library.

The assumption that the sequence of the first l-mer is known is well justified since the PCR reaction (Saiki et al., 1988) is presently most often used to amplify (and often also to label) a DNA to be sequenced. (In PCR reactions oligonucleotide primers of a known sequence are used.) One can separate an amplified duplex DNA and use an appropriately labelled single-stranded DNA in SBH experiments. A length n of DNA used in SBH can be estimated using gel electrophoresis and length standards, and we assume its knowledge as a problem parameter.

It follows that a computational phase of the DNA sequencing method (SBH) should take into account the possibility of positive and negative error occurrence. Our approach is based on an idea that when reconstructing a DNA sequence, a maximum cardinality subset of l-mers from *spectrum* should be used. Moreover, a number of nucleotides in the obtained sequence should not be greater than in the source sequence. Joining two l-mers into a sequence is connected with a cost. Two l-mers of length l may overlap on $l-1$, $l-2$, ..., or 0 nucleotides. The cost of joining two l-mers is equal to l minus a number of nucleotides that overlap in these l-mers. For example, two l-mers AATCG and CGGTG may overlap on two nucleotides and create a longer sequence AATCGGTG. Thus, a cost of joining them is equal to 3. We assume that l-mers are always maximally overlapped but later the final sequence can be lengthened if its length is less than n, by allowing less than maximal overlap between neighbors in the sequence. In this way a number of successive sequences may be derived from the one obtained during the branch and bound process, but all of them have the same arrangement of oligonucleotides.

In the next Section the basic method and its two variants will be presented, depending on assumptions about the SBH experiment. Three methods distinguished are: the general case with positive and negative errors; only negative errors; and only positive errors, respectively. Then Section 3 presents computational results.

2 The Basic Method and Its Variants

As we discussed, the problem to be considered may be formulated as follows. Given *spectrum* (a set of l-mers that hybridize to an analyzed sequence N) and length n of sequence N, reconstruct the latter. The *spectrum* may contain negative and positive errors. We propose to formulate this problem as a variant of a *Selective Traveling Salesman Problem* (STSP). In a classical *Traveling Salesman Problem* (TSP) there is given a complete directed or undirected graph where to each arc or edge a cost is assigned. The goal is to visit every vertex in the graph exactly once and return to the starting point in such a way that a sum of costs of traversed arcs or edges included into the cycle is minimal. This problem is known to be strongly NP-hard, thus unlikely to admit a polynomial-time algorithm (Garey and Johnson, 1979).

The directed STSP is a modification of the classical version. In this problem, in addition to arc costs, there is defined a profit for each vertex. The goal is to maximize a total profit of the visited vertices while the total cost of traversed arcs should not exceed a given value. In this version of the problem the first vertex must be also the last one.

If in the STSP the path is not required to be a cycle and each vertex profit is equal to 1, then the new problem may be shown to be equivalent to the considered DNA sequencing problem. In this case each l-mer from the *spectrum* corresponds to a vertex in the graph. Each arc incident with two vertices in the graph is labeled by an overlapping cost of the two l-mers corresponding to these vertices. One should notice that this cost depends on the order in which these two l-mers are joined. Hence the graph must be directed. Moreover, there are no loops in this graph. Note, that in a case when a path is a cycle this method can be used to sequence circular DNA strands.

We see that the above formulation can handle positive as well as negative errors, the nature of which is irrelevant for the method proposed. Positive errors are allowed by assuming the possibility that $|spectrum| > n - l + 1$. On the other hand, negative errors are also taken into account. In case when two l-mers are joined with cost greater than 1 additional l-mers (not present in *spectrum*) are included between them. These additional l-mers are in fact negative errors. For example joining l-mers AATCG and CGGTG produces longer sequence AATCGGTG. In this longer sequence we may distinguish two additional l-mers of length 5: ATCGG and TCGGT.

Constructing a desired path in the above variant of *Selective Traveling Salesman Problem* is equivalent to finding an unknown sequence N composed of a maximum cardinality subset of l-mers from *spectrum*. This modified version of STSP may be formulated as the following *Integer Linear Programming* (ILP) problem. Here, we use some ideas presented in (Laporte and Martello, 1990). The formulation below as well as algorithms used for solving the problem take into account the specificity of the DNA sequencing problem, and thus they are different from those given in the above paper. The ILP problem may now be stated as follows:

maximize

$$p_1 + \sum_{i=2}^{z} p_i \sum_{j=1}^{z} x_{ji} \tag{1}$$

subject to

$$\sum_{i=1}^{z} x_{1i} = 1 \tag{2}$$

$$\sum_{i=1}^{z} x_{i1} = 0 \tag{3}$$

$$\sum_{i=1}^{z} x_{ki} \leq 1, \ k = 2, \dots, z \tag{4}$$

$$\sum_{i=1}^{z} x_{ik} \leq 1, \ k = 2, \dots, z \tag{5}$$

$$\sum_{i=1}^{z} x_{ki} - \sum_{j=1}^{z} x_{jk} \leq 0, \ k = 2, \dots, z \tag{6}$$

$$\sum_{i=1}^{z} \sum_{j=1}^{z} c_{ij} x_{ij} \leq c \tag{7}$$

$$\sum_{v_k \in S} (\sum_{i=1}^{z} x_{ik} + \sum_{j=1}^{z} x_{kj}) \leq 2|S| \sum_{v_i \in V \setminus S, v_j \in S} x_{ij}, \ \ S \subset V \setminus \{v_1\} \tag{8}$$

where:

z - a cardinality of *spectrum*,
V - a set of vertices (i.e. l-mers) of cardinality z,
$p_i = 1$ - a profit connected with vertex i (the same for each l-mer),
c_{ij} - a cost of an arc joining vertex i with vertex j,
$c = n - l$ - a maximum allowed cost,
x_{ij} - a boolean variable; it is equal to 1 if an arc joining vertices i and j is included in the solution; otherwise it is equal to 0.

We see that the criterion function (1) to be maximized is equivalent to a number of vertices selected in the *traveling salesman* path (i.e. l-mers constituting sequence N). The first vertex (i.e. l-mer) of the path is assumed to be known. (Later we will show how to omit this condition.) In the above formulation, the first vertex is denoted by variables with index 1. Equation (2) guarantees that the starting vertex (l-mer) has only one immediate successor in the solution. Similarly, equation (3) guarantees that in the solution there is no predecessor of the starting vertex (l-mer). Inequalities (4) and (5) ensure that each vertex included in the solution, except for the starting one, has at most one immediate successor and one immediate predecessor in the solution path, respectively. According to inequalities (6) there is only one starting point in the path. Because of constraint (7) a cost connected with the solution is not greater than a given value (equal to $n - l$). Inequality (8) ensures that there is no separate cycle in the solution.

To solve the above problem a branch and bound type algorithm is proposed. Starting from the first vertex (l-mer), a solution is constructed gradually by adding consecutive vertices. At a current vertex h an upper bound for the possible extensions of the sequence is calculated according to the following ILP formulation:

maximize

$$P(h) + \sum_{j \in H(h)} p_j y_j \tag{9}$$

subject to

$$C(h) + \sum_{j \in H(h)} y_j \Big(\min_{i \in H(h) \cup \{h\} \setminus \{j\}} \{c_{ij}\} \Big) \leq c \tag{10}$$

where:

h - a current vertex,
y_j - a boolean variable; it is equal to 1 if vertex j may be included in the solution; otherwise it is equal to 0,
$H(h)$ - a set of unvisited yet vertices (l-mers),
$C(h)$ - a cost of a current sequence,
$P(h)$ - a number of l-mers already included into the current path.

In this way it is also possible to model instances of the sequencing problem where there are only positive errors, where an input graph is not a complete graph. The arcs join only such vertices that correspond to l-mers overlapped on $l - 1$ nucleotides. Note, that in this case all arc costs are equal to 1. In case where only negative errors appear, the input graph is the same as in the most general case, but there is one additional constraint: all l-mers from *spectrum* must be included into the sequence.

The general algorithm (*algorithm A*) accepting both types of errors, is presented below. The required length of sequence (n), the set of oligonucleotides (*spectrum*) and one element from *spectrum* assumed to be an initial one, are known at the beginning of the search process. One should obtain final sequences, each of them of length no longer than n, consisting of a maximum number of oligonucleotides from *spectrum*. A search tree, which is built during this process, has elements of *spectrum* (l-mers) as nodes. A path consisting of visited nodes in the tree corresponds to an arrangement of l-mers included into the sequence. The path begins with an initial oligonucleotide and is built until it becomes an acceptable solution, or until further branching is not feasible. Each solution with a maximum number of vertices is stored. The algorithm does not need any knowledge of an initial l-mer. If such an l-mer is not indicated, the algorithm successively treats each oligonucleotide as an initial one. In this case each of the l-mers becomes a root of a subtree built by the algorithm.

Nodes of the tree are visited according to *depth first search* method. During a construction of a path of l-mers, the procedure counts a cost connected with this path. If at a certain node in the search tree the cost of a current path exceeds the total feasible cost $n-l$, and the number of visited nodes is less than a current maximum, further branching is not allowed. Similarly, if in a current node the cost of the path does not exceed the total cost, but going further down from this node will not produce an acceptable solution, the node is not branched. This fact is discovered by counting a maximum number of non-visited nodes (criterion function (9)), that could be included into the path without violating the total cost (constraint (10)). This estimation often exceeds a number of vertices in an optimal solution, but it is a good upper bound for the discussed branch and bound method.

Successors of each node (all l-mers not yet visited) are ordered by nondecreasing cost. Hence, if after adding one of them into the path, further branching is not feasible because of exceeding the total cost, next successors on the list are rejected automatically. Moreover, such an ordering leads to an early construction of a good solution, that in turn results in more cuts in the search tree. The successor is omitted during branching if its inclusion into the path causes only duplication of an already obtained solution.

At the beginning of computation there are no nodes included in the path (i.e. to yield an actual value of the criterion function). It follows that a great part of the computation time may be wasted in finding bad, non-satisfactory solutions. To avoid that a heuristic algorithm has been proposed, that gives a good initial solution for the discussed branch and bound method. The longer the initial solution (i.e. a constructed DNA sequence) is, the greater the number of cuts in the search tree. The heuristic proposed is a greedy algorithm. It starts at initial oligonucleotide and adds successive l-mers while cost of the path is not greater than a total feasible cost. An l-mer is included in the path if a cost of overlapping of a current l-mer and the new one, plus the smallest cost of overlapping of the new one and one of its successors, is the smallest among other possible connections derived from the current node. This heuristic, although simple, generates a sequence of length usually close to n (Błażewicz et al., 1997a).

The algorithm described above constructs DNA sequences in the presence of both types of errors, negative and positive. However, it is also possible to pose a more theoretical question. If one had an additional knowledge about the nature of errors (i.e. only positive or only negative), what would be the gain in computation times? As one may easily guess (and this guess will be later supported by the results of a computational simulation), the gain is significant. Another question is, however, an experimental justification of this knowledge.

In the case where there are no positive errors, it is possible to find all solutions, not only those with a maximum value of the criterion function. Each of the solutions includes all elements of *spectrum* (not only a part of them, as is the case of the most general method). Each permutation of these elements that forms a chain of nucleotides of length not greater than n is a feasible solution. An algorithm searching for such chains (*algorithm B*) is based on a branch and bound method too. A search tree on which this algorithm operates has the same structure as the one which is used by the general algorithm. In the case when a number of errors (which is known and equal to $n-l+1-|spectrum|$) is not greater than $l-2$, a set of successors of each node can be restricted. l-mers being successors of the given node with a connection cost greater than a number of errors plus 1, are deleted from this set. If the number of errors is greater than $l-2$ one cannot restrict sets of succesors, because in this case each l-mer may be a successor of each other. Moreover, in each node the algorithm checks whether or not it is possible to obtain a feasible solution in a subtree rooted in this node. If an upper bound on a number of l-mers that can be included into a solution is different from $|spectrum|$, a branching is not feasible. The procedure that computes an upper bound is similar to the one from the most general algorithm A. If all elements of *spectrum* are included into the path and its cost does not exceed a feasible cost, this path is stored as one of the solutions.

Yet another simplification of the general algorithm is possible. In case a number of negative errors does not exceed $l-2$, each l-mer from *spectrum* has its own counter of predecessors. At the beginning of the search process each of these counters has a value equal to a number of l-mers for which the given l-mer can become an immediate successor, e.g. a number of l-mers that cost of joining with the given l-mer is not greater than a number of errors plus 1. During computations all counters are updated (i.e. decreased if a predecessor is added to a generated sequence). If in any node of the search tree one of the counters has its value equal to zero, the only feasible way of branching this node is the one which leads to an l-mer with zero-valued counter as an imediate successor of the current node. Any other way of branching makes impossible an inclusion of this l-mer into a path and the algorithm does not give a feasible solution.

For the case where there are only positive errors, another method (*algorithm C*), much simpler than algorithm A, has been proposed. It is also based on the branch and bound method. In this case the arcs with cost equal to 1 are only considered. Greater cost would indicate a negative error. Note, that in this case our graph contains many fewer arcs than in the general case. The method constructs sequences of oligonucleotides of length equal exactly to $n-l+1$ (i.e. paths of cost equal to $n-l$), and it also finds all feasible solutions. In the case when a starting l-mer is not known, each of *spectrum*'s element becomes a root of a subtree, similarly as in the previous method. During the search process the algorithm does not compute an upper bound, nevertheless the method is very efficient. This efficiency is caused by a structure of the search tree which consists only of arcs labeled by a cost equal to 1.

In the next Section the results of computational tests of the described algorithms are reported.

3 Computational Results

The described algorithms have been tested by an extensive computational experiment. In these tests DNA sequences obtained from GenBank, National Institute of Health, USA, have been used. DNA coding sequences, further denoted by the symbols St001-St109*, have been chosen. They are human mRNA for various proteins. All computational experiments have been carried out on SGI Power Challenge in Poznań Supercomputing and Networking Center. The algorithms have been implemented using ANSI C language.

The data for the experiment have been prepared in the following way. From the source sequences taken from GenBank, initial strings of length about $100-800$ nucleotides have been extracted to obtain *spectra* consisting of $100-800$ elements, respectively. (Exact lengths of extracted initial strings depend on the length l of hybridizing oligonucleotide.) Next, a hybridization experiment has been simulated by reading all oligonucleotides of length 10 from these sequences. The lengths of long strings (n) and oligonucleotides (l) have been chosen on the basis of real biochemical experiments (Caviani Pease et al., 1994). Moreover, a length of oligonucleotides equal to 10 gives high probability that one gets as a solution only one DNA sequence. (The dependence of an average number of generated sequences on n and l, has been presented in (Błażewicz et al., 1997b).) This fact has been confirmed by tests - in all cases where $|spectrum| < 500$ exactly one solution, identical with a source sequence, has been obtained. What is more, for the value $l = 10$, a number of negative errors caused by oligonucleotide repetitions is relatively small even in real DNA sequences (Błażewicz et al., 1997a). The algorithms presented have been adapted to any values of n and l ($l \leq n$)

*The list of corresponding accession numbers in GenBank database is available by request from the authors.

and generate a full list of feasible solutions (according to the assumptions given for particular algorithms).

In the experiment a set of oligonucleotides from a source sequence and sorted alphabetically is a *spectrum*. (Of course the algorithm would run on an arbitrary set of oligonucleotides.) In all cases a starting oligonucleotide is known, unless stated otherwise.

The following idea has been used to introduce *errors* into the *spectrum*. In what follows a notation "100-10+10" means that from an ideal *spectrum* (without any errors) of cardinality 100, 10 *randomly* selected (according to a uniform distribution) l-mers (*negative errors*) have been deleted, and next 10 *randomly* generated *positive errors* have been included. Moreover, in all cases a starting l-mer is not deleted and l-mers which have been included in *spectrum* had to be different from those already existing in it. Negative errors have been generated gradually. Firstly, from an ideal *spectrum* 10 l-mers have been deleted and a set "100-10" has been created. From this set, by a subtraction of next 10 l-mers, a set "100-20" has been created etc. Similarly, by an addition of 10 l-mers to an ideal *spectrum* a set "100+10" has been created, an addition of next 10 l-mers created a set "100+20", etc. A set denoted by "100-10+10" has been created by an addition of 10 l-mers to the set "100-10". They were the same 10 l-mers being positive errors in the set "100+10". Three algorithms A, B, C have been tested, where A, B, C denote respectively, the general algorithm, its version for negative errors only and the version for positive errors only.

At the beginning a series of tests have been made to establish *an importance of the heuristic algorithm* used for generating a starting solution (i.e. a lower bound on the value of the optimal solution). Results (presented in (Błażewicz et al., 1997a)) have shown that the heuristic algorithm causes a significant reduction of the search tree and thus a search time in the worst cases. (It should be noted, however, that it does not find an exact solution in majority of the cases.)

A goal of the next series of experiments was a comparison of computation times of all the algorithms A, B and C, for the same sets of data. Obviously, these tests could have been carried out only for *spectra* without errors. For these tests 10 DNA strings have been selected (St001 $-$ St010). In order to obtain *spectra* consisting of $100-800$ elements respectively, $109-809$ initial nucleotides from these strings have been taken and sets of overlapping 10-mers formed. Table 1 contains average computation times depending on the cardinalities of *spectra*. Each entry shows an average computation time of 10 instances (taking into account only those instances which have been solved within 2500 seconds). The number of solved instances is shown below the average time.

For the same above 10 DNA strings and the same value of l, a comparison of pairs of algorithms A, B and A, C, respectively, has been carried out. In these cases negative errors for the first pair and positive errors for the second one, respectively, have been introduced. Tables 2 and 3 contain a comparison of average computation times for the respective pairs of algorithms, and numbers of instances solved within 2500 seconds. As we see, the relatively very good behavior of algorithms B and C heavily depends on the additional knowledge of the type of errors present in *spectrum*.

Both types of errors could only be handled by algorithm A. Table 4 shows the results of tests of this algorithm in the presence of both types of errors.

Since *negative errors* appeared to be more time consuming, another set of tests have been carried out to check an influence of the source of errors on computation times. Firstly, *natural repetitions of oligonucleotides* in DNA sequences have been assumed to create negative errors. In order to achieve this goal a larger set of instances for testing had to be used. The parameters of the new experiment have been chosen in such a way that a diversity of numbers of negative errors was as large as possible. The values $n = 105$ and $l = 6$ have been chosen. Numbers of instances (from among 109, St001-St109) with a given number of negative errors used in this experiment are shown in Table 5. In Table 6 are shown average computation times of algorithm B for DNA strings with numbers of errors depicted in Table 5 (here no limit for computation time has been assumed). In this series of computations there was no l-mer indicated as the starting one, so computation times are respectively greater.

Secondly, an *experimental source* of negative errors has been assumed. To simulate these errors they have been generated randomly. Among 109 DNA strings there were 13 strings without natural repetitions of oligonucleotides for $n = 105$ and $l = 6$ (strings denoted by St012 - St024). From these strings 1 to 8 l-mers have been randomly removed. Results of this experiment for the same algorithm B are shown in Table 7. One can notice on the basis of Tables 6 and 7 that natural repetitions in *spectrum* make computational times much longer. In order to minimize a number of such repetitions one should choose proper values of n and l.

On the other hand, we have also tried to test the algorithm for a *different model of positive error appearance* (not only generated randomly). In order to include one false element into a *spectrum*, one of the l-mers already existing in it has been duplicated and then its first, last or both nucleotides have been changed. Because for this kind of errors results were almost identical to those obtained earlier (the difference was not greater than 0.01 second), more complicated input data have been prepared. The worst type of errors for the described algorithm is the one where the last nucleotide of an l-mer is false. A series of tests with only this type of error has been carried out. Again results were very similar to those obtained for the randomly generated errors.

4 Conclusions

In the paper, the original algorithm (including two its variants) for solving a computational phase of the DNA sequencing by hybridization method, has been proposed. The algorithm can handle the cases of negative and positive errors, appearing in data obtained from the hybridization experiment. No a priori knowledge about the nature nor rates of these errors is required. On the other hand, if a certain type of error can be excluded, then the computation times of the algorithm (or rather its variants) can be much shorter.

The presented algorithms have been extensively tested on real, thus computationally harder, DNA sequences. Parameters n (the length of a reconstructed sequence) and l (the length of hybridizing oligonucleotides) chosen for test purposes, had the values used in real experiments. The tests have shown a good performance of the algorithm in the presence of both types of errors. The algorithm has behaved exceptionally well if only positive errors have appeared (and this knowledge could be used). The general case, where positive and negative errors occurred could be also solved satisfactorily for an error rate achieving 10%. The worst situation of the algorithm's performance appeared to be for the case of negative errors caused by repetitions of l-mers in a DNA sequence. This obstacle could be eliminated in two ways. Firstly, by a proper selection of hybridization parameters l versus n. Secondly, by choosing such conditions of the hybridization experiment that would tend to eliminate negative errors at the expense of introducing additional positive ones.

Future examinations should allow for a further evaluation of the scope of applications of the sequencing approach with positive and negative errors described in this paper.

Acknowledgment

The authors gratefully acknowledge helpful remarks made by Mike Waterman and the referee.

References

Bains, W. 1991. Hybridization methods for DNA sequencing. *Genomics* 10, 294-301.

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

Błażewicz, J., Formanowicz, P., Kasprzak, M., Markiewicz, W.T., Węglarz, J. 1997a. DNA sequencing with errors. *Report RA-002/97.* Poznań Supercomputing and Networking Center.

Błażewicz, J., Kaczmarek, J., Kasprzak, M., Markiewicz, W.T., Węglarz, J. 1997b. Sequential and parallel algorithms for DNA sequencing. *CABIOS* 13, 151-158.

Błażewicz, J., Kasprzak, M., Sterna, M., Węglarz, J. 1996. Selected combinatorial optimization problems arising in molecular biology. *Ricerca Operativa* 26, 35-63.

Caviani Pease, A., Solas, D., Sullivan, E.J., Cronin, M.T., Holmes, C.P., Fodor, S.P.A. 1994. Light-generated oligonucleotide arrays for rapid DNA sequence analysis, *Proc. Natl Acad. Sci. USA* 91, 5022-5026.

Chee, M., Cronin, M.T., Fodor, S.P.A., Gingeras, T.R., Huang, X.C., Hubbell, E.A., Lipshutz, R.J., Lobban, P.E., Miyada, C.G., Morris, M.S., Shah, N., Sheldon, E.L., Fodor, S.P. 1995. New arrays of oligonucleotide probes - used for comparing known sequences with variants for detection of mutation(s) and sequencing. Affymax Technologies NV. USA:WO 9511995.

Drmanac, R., Labat, I., Brukner, I., Crkvenjakov, R. 1989. Sequencing of megabase plus DNA by hybridization: theory and the method. *Genomics* 4, 114-128.

Drmanac, R., Labat, I., Crkvenjakov, R. 1991. An algorithm for the DNA sequence generation from k-tuple word contents of the minimal number of random fragments. *J. Biomol. Struct. Dyn.* 8, 1085-1102.

Fodor, S.P.A., Read, J.L., Pirrung, M.C., Stryer, L., Lu, A.T., Solas, D. 1991. Light-directed, spatially addressable parallel chemical synthesis. *Science* 251, 767-773.

Fodor, S.P.A., Goldberg, M.J., Goss, V., Mcgall, G., Pease, R.F., Rava, R.P., Stryer, L., Winkler, J.L. 1996. Forming polymers having diverse monomer sequences on substrate - where substrate comprises linker mol. and protective gp., applying barrier layer and exposing regions of linker mol. layer to vapour comprising deprotecting agent. Affymax Technologies NV.EP 728520.

Gallant, J., Maier, D., Storer, J.A. 1980. On finding minimal length superstrings. *J. Comput. Syst. Sci.* 20, 50-58.

Gallant, J.K. 1983. The complexity of the overlap method for sequencing biopolymers. *J. Theor. Biol.* 101, 1-17.

Garey, M.R., Johnson, D.S. 1979. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W.H. Freeman and Co., San Francisco.

Guenoche, A. 1992. Can we recover a sequence, just knowing all its subsequences of given length? *CABIOS* 8, 569-574.

Hacia, J.G., Brody, L.C., Chee, M.S., Fodor, S.P.A., Collins, F.S. 1996. Detection of heterozygous mutations in BRCA1 using high density oligonucleotide arrays and two-colour fluorescence analysis. *Nature Genet.* 14, 441-447.

Khorlin, K.K., Khrapko, K.R., Lysov, Y.P., Yershov, G.K., Vasilenko, S.K., Florentiev, V.L., Mirzabekov, A.D. 1991. An oligonucleotide matrix hybridization approach to DNA sequencing. *Nucleic Acids Res.* 24, 191-192.

Khrapko, K.R., Lysov, Y.P., Khorlin, A.A., Shik, V.V., Florentiev, V.L., Mirzabekov, A.D. 1989. An oligonucleotide approach to DNA sequencing. *PEBS Letters* 256, 118-122.

Laporte, G., Martello, S. 1990. The selective travelling salesman problem. *Discrete Appl. Math.* 26, 193-207.

Lipshutz, R.J. 1993. Likelihood DNA sequencing by hybridization. *J. Biomol. Struct. Dyn.* 11, 637-653.

Lipshutz, R.J., Morris, D., Chee, M., Hubbell, E., Kozal, M.J., Shah, N., Shen, N., Yang, R., Fodor, S.P.A. 1995. Using oligonucleotide probe arrays to access genetic diversity. *BioTechniques* 19, 442-447.

Markiewicz, W.T., Andrych-Rożek, K., Markiewicz, M., Żebrowska, A., Astriab, A. 1994. Synthesis of oligonucleotides permanently linked with solid supports for use as synthetic oligonucleotide combinatorial libraries. Innovations in solid phase synthesis, 339-346. *In* Epton, R., ed., *Biological and Biomedical Applications*, Mayflower Worldwide.

Maskos, U., Southern, E.M. 1993. A study of oligonucleotide reassociation using large arrays of oligonucleotides synthesised on a glass support. *Nucleic Acids Res.* 21, 4663-4669.

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

Pevzner, P.A., Waterman M.S. 1995. Open combinatorial problems in computational molecular biology. *Proceedings of Israel Symposium on the Theory of Computing and Systems*, Tel Aviv, 158-173.

Saiki, R.K., Gelfand, D.H., Stoffel, S., Scharf, S.J., Higuchi, R., Horn, G.T., Mullis, K.B., Erlich, H. 1988. Primer-directed enzymatic amplification of DNA with thermostable DNA polymerase. *Science* 239, 487-491.

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

Waterman, M.S. 1995 *Introduction to Computational Biology. Maps, Sequences and Genomes.* Chapman & Hall, London.

Algorithm	*spectrum*							
	100	200	300	400	500	600	700	800
A	0.64	30.21	21.05	82.61	405.74	273.00	547.85	738.94
	10	10	9	9	9	7	7	5
B	0.04	0.15	0.33	0.60	0.97	2.54	20.42	2.66
	10	10	10	10	10	10	10	9
C	0.01	0.05	0.11	0.19	0.31	0.44	0.73	0.91
	10	10	10	10	10	10	10	10

Table 1. Average computation times for algorithms A, B, C and no errors (above) and numbers of instances solved within 2500 seconds time bound (below).

Algorithm	*spectrum*							
	100-10	200-10	300-10	400-10	100-20	200-20	300-20	400-20
A	0.72	3.97	24.11	323.77	3.32	229.48	157.75	389.71
	10	9	8	8	10	10	7	8
B	0.12	1.39	3.90	10.31	0.14	1.91	7.67	20.22
	10	10	10	10	10	10	10	10
Algorithm	*spectrum*							
	100-30	200-30	300-30	400-30	100-40	200-40	300-40	400-40
A	3.12	2.84	171.99	269.99	2.04	2.47	51.99	30.67
	10	8	7	7	10	9	7	5
B	0.11	1.66	7.92	21.54	0.08	1.42	7.26	19.44
	10	10	10	10	10	10	10	10

Table 2. A comparison of average computation times for algorithms A and B (above) and numbers of instances solved within 2500 seconds time bound (below).

Algorithm	*spectrum*							
	100+20	200+20	300+20	400+20	100+40	200+40	300+40	400+40
A	1.16	42.13	28.07	96.47	1.97	55.64	35.73	117.03
	10	10	9	9	10	10	9	9
C	0.02	0.06	0.12	0.21	0.02	0.07	0.14	0.23
	10	10	10	10	10	10	10	10
Algorithm	*spectrum*							
	100+60	200+60	300+60	400+60	100+80	200+80	300+80	400+80
A	3.25	70.41	43.39	130.89	4.71	87.96	52.66	163.16
	10	10	9	9	10	10	9	9
C	0.03	0.08	0.16	0.25	0.04	0.09	0.17	0.28
	10	10	10	10	10	10	10	10

Table 3. A comparison of average computation times for algorithms A and C (above) and numbers of instances solved within 2500 seconds time bound (below).

Algorithm	*spectrum*			
	100-10+10	200-10+10	300-10+10	400-10+10
A	1.05	4.66	29.33	385.02
	10	9	8	8
	spectrum			
	100-20+20	200-20+20	300-20+20	400-20+20
	7.73	4.66	261.36	537.24
	10	9	7	8
	spectrum			
	100-30+30	200-30+30	300-30+30	400-30+30
	11.88	4.66	369.16	52.53
	10	8	7	6
	spectrum			
	100-40+40	200-40+40	300-40+40	400-40+40
	177.20	4.63	141.47	46.27
	10	9	7	5

Table 4. Average computation times for algorithm A and the instances with positive and negative errors (above) and numbers of instances solved within 2500 seconds time bound (below) .

Algorithm	*spectrum*								
	100	100-1	100-2	100-3	100-4	100-5	100-6	100-7	100-8
B	13	16	11	10	14	10	4	5	2

Table 5. Numbers of instances (from among 109) with a given number of negative errors caused by oligonucleotide repetitions.

Algorithm	*spectrum*								
	100	100-1	100-2	100-3	100-4	100-5	100-6	100-7	100-8
B	0.13	1.41	2.22	119.68	197.01	627.30	2965.86	9386.69	11505.02

Table 6. Average computation times of algorithm B for *spectra* with repetitive negative errors.

Algorithm	*spectrum*								
	100	100-1	100-2	100-3	100-4	100-5	100-6	100-7	100-8
B	0.13	0.26	0.26	0.42	1.27	3.95	7.23	10.07	11.81

Table 7. Average computation times of algorithm B for *spectra* with randomly generated negative errors.