Sequential and parallel algorithms for DNA sequencing

Jacek Blazewicz¹,⁴, Janusz Kaczmarek¹, Marta Kasprzak¹, Wojciech T. Markiewicz² and Jan Weglarz¹,³

Abstract

Motivation: Reconstruction of the original DNA sequence in the sequencing by the hybridization approach (SBH) requires computational support due to a large number of possible combinations. One can notice a lack of algorithms admitting false-negative data and giving in addition all possible solutions.

Results: In this paper, a new method of sequencing has been proposed. An algorithm based on its idea (for the general case, when some data are missing, like in the real experiment) has been implemented and tested. Authentic DNA sequences have been used for testing. A parallel version of the algorithm has also been implemented and tested. The quality of the reconstruction is satisfactory for the library of oligonucleotides of length between 8 and 12, and 100, 200 and 300 bp long sequences. A way to a further decrease in the computation time is also suggested.

Availability: The program is available on the EMBL file server.

Contact: E-mail: blazewicz@poznlv.put.poznan.pl

Introduction

The recognition of DNA sequences, as part of a wide stream of activities which aim at completing our knowledge about the biochemical basis of life, attracts researchers from various domains. Three main fields of activities can be distinguished in this area: DNA sequencing—aimed at discovering the exact sequence of nucleotides in a relatively short DNA segment (100–300 bp long); DNA assembling—which assembles the sequenced fragments into longer (1000–10 000 bp long) contigs; DNA mapping—where one deals with whole chromosomes and tries to place marked DNA fragments (usually genes) on a certain chromosome region. This paper deals with DNA sequencing. Our method is based on the hybridization experiment, several versions of which have been proposed, e.g. by Drmanac et al. (1989), Khrapko et al. (1989), Southern et al. (1992) and Markiewicz et al. (1994).

Before presenting the method, let us set up the problem more precisely. In general, the hybridization experiment is based on the property of single-stranded nucleic acids of forming a complex (antiparallel) with a complementary strand of a nucleic acid. Shorter fragments of nucleic acids (oligonucleotides) are used in hybridization experiments and, thus, the formation of the complex indicates the occurrence of a sequence complementary to the oligonucleotide in the nucleic acid. As a result of the experiment, one gets a set S of all l-long oligonucleotides which are known to hybridize with the investigated DNA sequence N of length n (i.e. they are substrings of string N). In the case of ideal data, we thus have |S| = 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 (e.g. it can contain repeated subsequences). In this case, negative error occurs. On the other hand, we assume no positive errors, i.e. each fragment (oligonucleotide) present in S is a part of the original sequence. The goal is to reconstruct the original sequence from the given set S.

Until now, several methods for assembling the original sequence have been proposed, e.g. by Lysov et al. (1988), Khrapko et al. (1989), Pevzner (1989), Bains (1991) and Guénoche (1992). For the case of a complete Spectrum (i.e. no errors), Lysov suggests looking for a Hamiltonian path in a special graph (its vertices are elements of S, an edge between vertices exists if, and only if, the corresponding elements overlap on length l − 1). Since the problem of finding the Hamiltonian path is known to be NP-hard (thus, unlikely to admit polynomial time algorithms), Pevzner proposes an interesting transformation of such a graph. As a result of the transformation, one obtains a new graph in which a Eulerian path is looked for. This establishes the complexity of the problem since the problem of finding a Eulerian path can be solved in polynomial time. For the case of negative errors (when some data in S are missing, i.e. |S| = n − l + 1 − k, k > 0), Pevzner proposes another method for finding missing arcs in the above graph based on the transportation problem. However, as pointed out by Blazewicz et al. (1996), this method does not deal correctly with some cases and the DNA sequence cannot be reconstructed. On the other hand, Bains’s algorithm is most similar to the one proposed in the present paper, but does not take care of the missing data. The paper of Guénoche (1992) presents the implementation of the method.

¹Institute of Computing Science, Poznan University of Technology, ul. Piotrowa 3a, 60-965 Poznan and ²Institute of Bioorganic Chemistry, Polish Academy of Sciences, ul. Noskowskiego 12/14, 61-704 Poznan and ³Poznan Supercomputing and Networking Center, Poznan, Poland

*To whom correspondence should be addressed.
based on finding Hamiltonian paths. All the above algorithms are sequential approaches, whereas a significant speed-up can be obtained by applying parallel procedures.

In this paper, a new method for DNA sequencing is proposed. An algorithm based on its idea has been developed and implemented. It solves the more general case, when some data are missing (false-negative errors), like in the real experiment. A parallel version of the algorithm is also discussed.

The structure of the paper is as follows. In the next section, we present the sequential version of the method for cases when data are incomplete. A modification of the method for long \( l \) is also presented. In subsequent sections, sources of defects in the experimental data are discussed, and some remarks on the implementation, test results and their analysis are also included. The final section contains the description of the parallel implementation of the algorithm.

The method and its sequential implementation

The proposed method is rather a brute force one, but thanks to the structure of the data, it is expected to terminate fast in the case where set \( S \) is complete, i.e. \( |S| = n-l+1 \). In the case where it is not (i.e. \( |S| = n-l+1-k, k>0 \), where \( k \) is a number of oligonucleotides present in sequence \( N \) that are missing in \( S \)), the running time of the method is longer, but it always generates a correct sequence (in case only one is possible) or sequences.

The basic idea behind it is as follows. Suppose \( k = 0 \). Then a complete tree of all possible sequences of length \( n \) is started to be built. The tree is constructed in the depth-first search way. At each level \( m \), there are \( 4^m \) nodes, due to four nucleotides present in DNA. As soon as it is possible (at level \( l \)), one tries to cut off (not to construct further) those branches which contain fragments not appearing in \( S \). Computational experiments have shown that in the case when \( |S| = n-l+1 \) and the sequence can be reconstructed unambiguously (in some cases, although the data are complete, due to the data structure the original sequence cannot be reconstructed unambiguously) an average depth of the tree built (except for the correct branch) is close to \( l \). Suppose now that there are \( k \) defects in set \( S \), i.e. \( |S| = n-l+1-k \). The process of building the tree changes now (although the general idea of the method remains the same). The origin of the defects—whether introduced by repeated fragments or resulting from experimental errors—does not have any influence on the method. The described method deals correctly with any type of false-negative errors. When constructing the tree, one allows each branch to contain at most \( k \) fragments of length \( l \) not included in \( S \). Then each branch is checked against this property, starting at level \( l+k \).

The proposed method can be modified in order to shorten the computation time. Some branches of the tree can be cut off before they reach the level \( l+k \). At each level between \( k \) and \( l+k \), the generated fragment is being checked whether it can be a part of any element of \( S \). In other words, it is not necessary to wait until level \( l+k \) is reached, the superfluous branches can be cut off earlier.

Thus, one may implement this idea in the form of the algorithm below. Initially, the parameters of the procedure \( \text{Reconstruct} \) are as follows: \( S \) is the set of fragments obtained from the experiment, \( \text{Sequence} \) is the reconstructed sequence (initially empty), \( \text{level} \) is the current level in the tree (initially 1). The resulting sequence (or sequences, when the reconstruction is ambiguous) is given in the line \( \text{output} \). The parameter \( i \) is the number of accepted fragments not present in \( S \) (initially 0).

Algorithm

\[
\begin{align*}
\text{Reconstruct}(S, \text{Sequence}, \text{level}, i) & \quad \text{begin} \\
& \quad \text{for} \; c \; \text{in} \; \{\text{A}', \text{C}', \text{G}', \text{T}'\} \; \text{do} \\
& \quad \text{begin} \\
& \quad \text{Sequence} := \text{Sequence} + c; \\
& \quad \text{if} \; \text{level} > k \; \text{and} \; \text{level} < k + l \; \text{then} \\
& \quad \text{begin} \\
& \quad b := i; \\
& \quad \text{for} \; \text{each ending fragment of Sequence} \\
& \quad \text{of length } <= l \\
& \quad \text{begin} \\
& \quad \text{if fragment <> beginning fragment of any } x \; \text{in} \; S \\
& \quad \text{then} \\
& \quad b := b + 1; \\
& \quad \text{if} \; b > k \; \text{then} \\
& \quad \text{begin} \\
& \quad \text{Sequence} := \text{Sequence} - c; \\
& \quad \text{end} \\
& \quad \text{end} \\
& \quad \text{end} \\
& \quad \text{if} \; \text{level} < l \; \text{then} \\
& \quad \text{Reconstruct}(S, \text{Sequence}, \text{level} + 1, i); \\
& \quad \text{else} \\
& \quad \text{begin} \\
& \quad \text{if fragment of last } l \; \text{nucleotides not included in } S \; \text{then} \\
& \quad \text{in Sequence = any } x \; \text{in} \; S \; \text{then} \\
& \quad \text{if level} = n \; \text{then} \\
& \quad \text{output} \; (\text{Sequence}); \\
& \quad \text{else} \\
& \quad \text{Reconstruct} \; (S \; - \; x, \text{Sequence}, \text{level} + 1, i); \\
& \quad \text{else if} \; i < k \; \text{then} \\
& \quad \text{if level} = n \; \text{then} \\
& \quad \text{output} \; (\text{Sequence}); \\
& \quad \text{else} \\
& \quad \text{Reconstruct} \; (S, \text{Sequence}, \text{level} + 1, i + 1); \\
& \quad \text{end} \\
& \quad \text{Sequence} := \text{Sequence} - c; \\
\end{align*}
\]
Let us illustrate the algorithm with a simple example.

**Example**

Let us consider the case where the number $k$ of oligonucleotides missing in set $S$ is greater than 0 (cf. Figure 1). We assume $N = ACGTCAT, S = \{ACG, GTC, TCA, CAT\}, n = 7, l = 3, k = 1$. Let us now trace how two exemplary branches of the tree have been constructed. First let us notice that $k = 1$, e.g. one, and only one, oligonucleotide of length $l = 3$ may be inserted while a branch is constructed. Consider branch AACG. The oligonucleotide AAC may be accepted for reconstruction. Although it does not appear in set $S$, one is allowed to accept one oligonucleotide (since $k = 1$) not appearing in $S$. The introduction of such an oligonucleotide is marked as $\textcircled{1}$. Next, ACG is chosen from $S$ since it is the only oligonucleotide in $S$ which may be appended to the AAC. One is no longer allowed to choose any oligonucleotide not contained in $S$. Moreover, it is no longer possible to extend the branch AACG since no oligonucleotide from $S$ may be appended to this sequence. Now, consider branch ACGTCAT that is the only correct result of the reconstruction. First, CGT $\textcircled{1}$ has been appended to ACG as the missing oligonucleotide. Thus, the limit of oligonucleotides from the outside of $S$ has been exhausted. From now on, only oligonucleotides from $S$ may be appended to the constructed sequence. The following oligonucleotides have been appended in order: GTC, TCA, CAT resulting in the original sequence ACGTCAT. Other branches of the tree can also be taken into consideration, although the tree presented in the Figure 1 is not complete. 

Obviously, the most time-consuming part of the process is...
the construction of the tree down to level \( l (l + k \text{ in the case of } k \text{ defects}) \). Thus, in order to speed up the computation, the starting fragment of the sequence to be reconstructed should be known. In fact, it is often known, especially when it was used as a primer (or part of it) in polymerase chain reaction (PCR) amplification (Saiki et al., 1988).

**Sources of false-negative errors**

As mentioned before, there are two sources of false-negative errors. One type is introduced by repeated substrings in a sequenced fragment, while the other appears as the result of experimental errors. Repeats in a sequence entail defects in \( S \), because it is possible to get from the experiment only an appearance of a substring, not a number of repetitions of it. We assume that intensities in the hybridization experiments do not indicate how many times subsequences appear.

The frequency of the first type of errors can be estimated statistically. However, when analysing data from the GenBank we realized, firstly, that the number of 'bad' sequences, i.e. sequences which introduce the defect \( k > 0 \) as a result of repeated fragments, not exactly meets the formula presented by Southern et al. (1992). This formula describes the probability with which all oligonucleotides occur in the random sequence only once:

\[
\log p_r = \frac{-r(r - 1)}{2t}, \text{ if } r \ll t
\]

where \( r = n - l + 1 \) is the number of oligonucleotides, \( t = 4^i \).

The sequences we have used behave worse than the theoretical estimation would suggest. In Figure 2, the behaviour of real DNA sequences against theoretical estimation is compared. The upper three lines in Figure 2 illustrate the theoretical probability calculated from the above formula. The corresponding lower three series are calculated from real sequences. The real DNA sequences contain repeated fragments more frequently than one would expect on the basis of Southern's formula. It appears that in real hybridization experiments, defects of this type will occur quite often.

Errors of the negative type may result from the use of a limited set of \( l \)-oligonucleotides (incomplete set) to probe DNA in a hybridization experiment. Then the probing of DNA with arrays of \( l \)-oligonucleotides (sequencing chips), performed in order to limit the appearance of false-positive errors under rather stringent conditions, may result in negative experimental errors.

**Results of the computational experiment**

An experiment evaluating our method from the computational point of view has been carried out. We have tried to find the dependence between various parameters experimentally rather than performing a theoretical analysis. First, we have used real DNA fragments that are known to be less random than stochastically generated sequences. To test our method, we have used 100, 200 and 300 bp long fragments taken from 40 various real DNA sequences. All the sequences come from GenBank (a database of investigated DNA fragments maintained by the National Institute of Health); the accession numbers are given in the Appendix. From each sequence, corresponding data sets \( S \) (as the result of a theoretical hybridization experiment) for \( l \in [8, 12] \) have been generated. Such values of \( l \) were chosen after analysis of real achievements in this area (cf. Caviani Pease et al., 1994), although any value is acceptable by the algorithm from a theoretical point of view. Each point on the charts shown

---

**Fig. 3.** Computation time of the algorithm versus the length of the original sequence (\( n \)) and the length of oligonucleotides (\( l \)) (assuming \( k = 0 \)).

154
in Figure 3 is an average of 40 results obtained for various sequences. All the results presented have been obtained on SGI Power Challenge in the Poznan Supercomputing and Networking Center.

The first set of experiments has been conducted for the case of no defects, i.e. $k = 0$. Surprisingly, although the full tree size grows exponentially, the time used by the algorithm grows linearly only, because $k = 0$ limits the growth of branches. This is illustrated in Figure 3.

In the case of the existence of false-negative errors ($k > 0$) resulting both from repeated substrings and experimental errors, one could expect that the increasing $k$ would increase the time exponentially (with $k$). This is because the complete tree has to be constructed down to the level $l + k$ and no branches can be cut off earlier. Fragments with repeated substrings have been taken into account and additional errors (thus imitating experimental errors) have been introduced randomly as well. The experiment speaks in favour of the above hypothesis. The increase in computation time in such a situation is presented in Figure 4. Let us stress, however, that the running time of the algorithm again depends linearly only on $n$ (the length of a reconstructed DNA sequence).

Another interesting factor measured by us concerns the average value of the number of defects resulting from the structure of the sequence, i.e. $\hat{k}$. Of course, one is interested in possibly low $\hat{k}$, since the bigger $\hat{k}$ is, the longer branches in the tree one has to consider and the longer the computation time is. Figure 5 illustrates the dependence of $\hat{k}$ on different values of $n$ and $l$ (assuming no experimental errors). One can observe that by increasing the length $l$ of oligonucleotides, one obtains a smaller $\hat{k}$ of oligonucleotides missing in $S$. This should result in decreasing the average computation time. However, the observed behaviour of the average computation time is somewhat surprising. As illustrated in Figure 6, the average...
computation time (versus the length of oligonucleotides $l$ for various lengths $n$ of a reconstructed sequence) initially decreases, which corresponds to the decreasing $k$ (cf. Figure 5). The following increase in the computation time corresponds to the situation where the increasing length of oligonucleotides $l$ does not affect the value of $k$ any longer, but the depth of the tree to be built completely (down to the level $l + k$) still increases.

A very important factor in sequence reconstruction is ambiguity. One is usually interested in the selection of parameters ensuring unambiguous reconstruction and at the same time in limited experimental effort. Curves in Figure 7 depicting an average number of generated sequences indicate that oligonucleotides of length 8 result in the generation of several possible sequences, those of length 9 result in the generation of only a few sequences, while 10 base long oligonucleotides ensure a unique reconstruction in most cases.

The results of our experiments indicate that a suitable length of oligonucleotides in a library used in the hybridization experiment would be 8, but sometimes this can lead to an ambiguous reconstruction. In such a case, oligonucleotides of length 9 for 200 and even 300 base long sequences could be used.

**Parallel implementation of the method**

Parallel implementation of the method can be constructed on the basis of the sequential version described above. Below we present an outline of this strategy.

At a certain level $m$ of the tree, the computations of the algorithm are split among a number of processes. Splitting the work at level $m$ results in generating $4^m$ chunks which can be treated independently from each other. The value of $m$ should be less than $l$, otherwise most of the branches could already be
Sequential and parallel algorithms for DNA sequencing

cut off before reaching level \( m \). A suitable number of potentially identical (regarding their processing times) processes facilitates the proper load balancing of the parallel system (i.e. a dynamic assignment of processes to processors in such a way that the processing time of the whole job is minimized).

The load-balancing strategy we have used is very simple, but leads to satisfactory results. The whole computation process is controlled by a single process called master. It builds the tree down to level \( m \) and, after that, sends requests to slave processes. Each of the slave processes performs only one chunk of work at a time. Once it is ready with its task, it sends the results to the controlling master process. The master process sends the next piece of work to the free slave process if the computation has not yet been completed.

We have tested the parallel version of the algorithm in computational experiments. It has been developed using the PVM system (Geist et al., 1994) and implemented on SGI Power Challenge. For the evaluation of the experiment, we have not used a real distributed heterogeneous system, like usual PVM-based systems, because the estimation of the speed-up and efficiency would not be simple in that case. Instead, we have run several PVM processes on a single machine. Then the speed-up has been computed as the quotient of the CPU time of the sequential version of the algorithm divided by the maximum CPU time used by the slave processes plus the master process CPU time. We have not used the wall time to determine the speed-up because it is strongly connected with the current load of the machine. On a dedicated system, we expect the results to be similar to these obtained in simulation, assuming we can omit the communication time.

Fig. 8. Speed-up obtained in simulation of the parallel version of the algorithm.

Fig. 9. The efficiency of the parallel algorithm.
Figures 8 and 9 illustrate the results of the parallel simulation. Each point in Figures 8 and 9 represents an average value of 40 runs for different data sets. As one can see, the speed-up obtained is nearly linear and does not depend on the number of processes applied. Such a good result has been obtained due to the proposed load-balancing algorithm.

In Figure 9, the efficiency (defined as the quotient of the speed-up divided by the number of processes performing the task) of the parallel version of the algorithm is presented. It worsens with increasing number of processes, but still remains above 87%. This result was obtained by breaking the workload into a proper number of pieces, making efficient load balancing possible.

Conclusions
We have presented sequential and parallel implementations of a new method for DNA sequencing from the data that result from the hybridization experiment. The presented method finds all possible sequences that can be reconstructed from the provided data even if the data are incomplete. Having carried out a lot of computational experiments, a reasonable trade-off between experimental and computational effort and unambiguity of the reconstructed sequence has been pointed out.

Future work should concentrate on three subjects. Firstly, an efficient method dealing with ambiguously reconstructed sequences should be developed. An interaction with existing knowledge about DNA structures could be used at the first stage. Experimental hybridization feedback at the second stage cannot be omitted. Secondly, the implementation of the parallel version of the method in a real (heterogeneous, non-dedicated) environment should be investigated in detail. Thirdly, it is desired to design a method dealing with false-positive errors in the set of oligonucleotides.

References